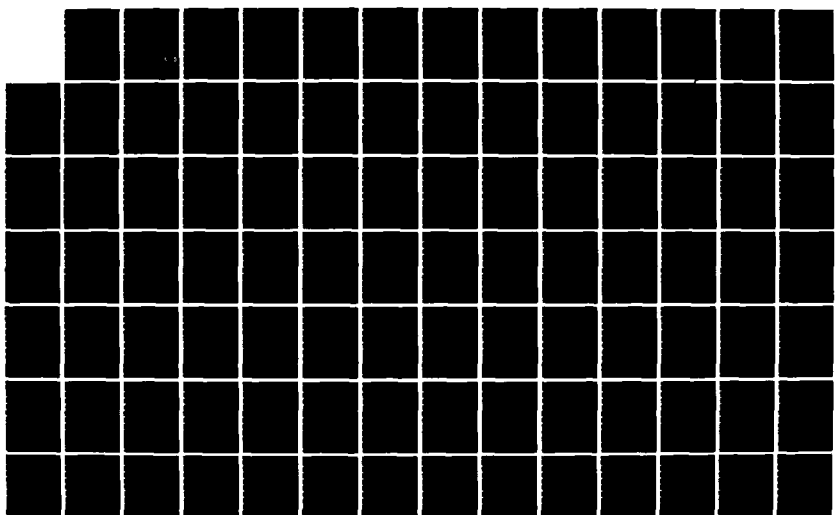
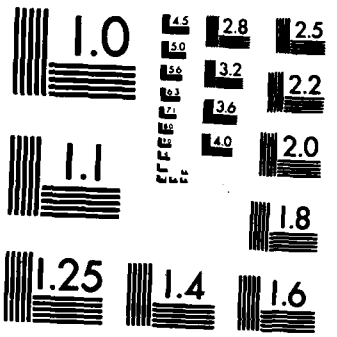
AD-A141 253 DEVELOPMENT OF AN AFIT (AIR FORCE INSTITUTE OF TECHNOLOGY) ADP SYSTEM NETWORK MODEL(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. 1/3

UNCLASSIFIED S M MCDERMOTT DEC 83 AFIT/GCS/OS/83D-1 F/G 9/2 NL





DEVELOPMENT OF AN AFIT ADP SYSTEM NETWORK MODEL

Steven M. McDermott, Captain, USAF

AFIT/GCS/OS/83D-1

AFIT/GCS/OS/83D-1

DEVELOPMENT OF AN AFIT ADP SYSTEM NETWORK MODEL

A Thesis

Presented to the Faculty of the School Of Engineering

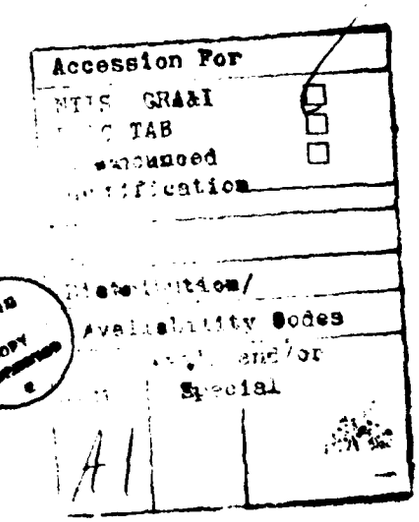
of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science



by

Steven M. McDermott, BS
Captain USAF

Graduate Computer Science
December 1983

Approved for public release;
distribution unlimited

PREFACE

This thesis reports on the development of an AFIT/ADP System Network Model. This system model consists of a workload and a network model. User's guides, maintenance guides, data dictionaries, and code listings are provided for each model in the appendices. After the system model is implemented it will be a powerful tool which ADP managers may use to manage the increasingly complex data processing system. The last chapter of this report outlines a progression of follow on efforts required to accomplish the implementation of this development.

I wish to express my sincere appreciation to Lt Col Thomas C. Clark, Jr., the advisor of this investigation, for his professional guidance, insight, and patience throughout the duration of this effort. I also wish to thank Thomas Hartrum, member of my thesis committee, for his support and patience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Data processing management is faced with more complex data processing configurations and with what configurations will be required in the future. As new devices become available, rapidly changing technologies coupled with different ratios of cost to performance are forcing management to review constantly the worth of their installed base of devices. Better management of resources can mean expenditures or savings of hundreds of thousands of dollars. Thus, decisions regarding the purchase of additional hardware will be under close scrutiny (Ref 10:52).

These decisions are not restricted to commercial companies but also must be dealt with by educational institutions such as the Air Force Institute of Technology (AFIT). In fact at an educational setting these decisions may even be more complex because of the varied requirements placed upon network resources. Many different compilers and application programs are required to teach and perform research. Often these programs are restricted to one type or size computer. Additionally, the number of users is usually quite large and dispersed across many buildings.

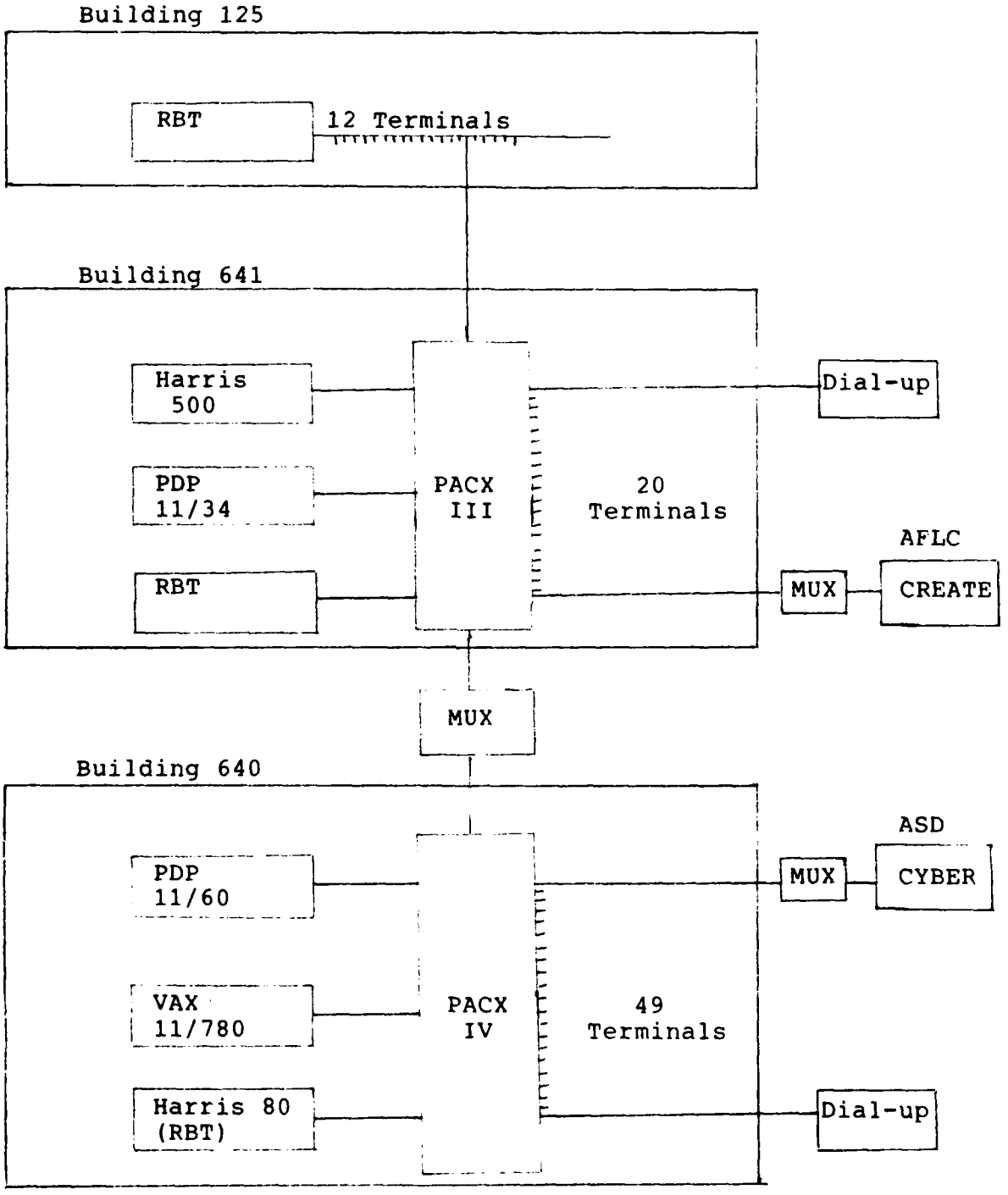
AFIT is currently going through an evaluation of the network. All of the schools and departments of AFIT have submitted a requirements/desire report to AFIT ACD. Using these reports ACD must evaluate many different hardware configurations (all within a specified budget) in order to

satisfy the majority if not all of these requirements. Coupled with these decisions is the uncertainty of the actual requirements after the acquisition of the hardware. No analytical tools currently exist to help ACD with this task. A network system model would assist ACD with these decisions. They then would be able to easily evaluate different hardware configurations with varied projected requirements.

The AFIT computer network consists of various computers and input/output devices interconnected through a GANDALF Private Automatic Communications Exchange (PACX) digital communication switch. (Figure 1.1). AFIT controlled resources are distributed between three physical locations while the CYBER and CREATE computers are located outside of AFIT. Every AFIT terminal connected to the PACX system can communicate with any of these computers. Dial-up capability to all computer resources is supported from both within AFIT and off facility.

AFIT's network began with only the CYBER computer with few terminals and remote batch locations. As each resource was added to the network, immediate saturation problems were elevated. However, in a short period of time the new resources also became saturated.

Future plans for this network include the addition of other computers, terminals, and software. As new resources are added, requirements on the old resources may shift to the new resources again relieving the saturation problem. However, at the same time new demands may be place upon the



RBT - Remote Batch Terminal

Figure 1.1 AFIT Network.

network. AFIT Automated Data Processing (ADP) managers are constantly trying to forecast the demands and acquire the additional resources to satisfy them.

The tools available to the ADP manager are judgement, intuition, experience, and analytical analysis of segments of the network system. The networks, however, have grown quite large and complex with many interrelationships between components of the system. The dynamics and interactions present such a complexity that it is impossible to model what is not understood. However, a simulation model is an attempt to gain a better understanding of the overall network system. A dynamic simulation model would provide ADP managers with a tool to use in addition to their intuition and experience in managing complex data processing systems.

Problem Statement

A dynamic model of the AFIT ADP network does not exist. A dynamic simulation model incorporating a hardware system structure with a characterization of the workload will enable ADP managers to study the effects and requirements of the continuously changing environment.

Research Question

What is the entire workload and network resources available to AFIT, how can the workload and resources be captured in a dynamic simulation model, and how can the model be used to evaluate policies, requirements, and added resources? Specifically:

1) What portion of the workload generated by AFIT utilizes network resources and how can this workload be translated into a dynamic workload model?

2) How can the network resources with their complex relationships be incorporated into a model that will enable ADP managers to evaluate the system given different hardware configurations and requirements?

Objectives

The primary objective of this research is to provide an initial validated dynamic system model of the AFIT ADP network. A dynamic model will enable ADP managers to study effects of policy changes, increased requirements (external changes), and enhancements to the system. Intermediate objectives are:

1) Develop independent workload and network models which when combined will form the system model.

2) Verify and validate all three models.

3) Provide guidance on how to use and alter all three models.

Scope

This research is directed at determining the portion of the AFIT ADP workload which uses AFIT ADP network resources. Specifically, the system model will determine turnaround times, response times, and utilizations of each ADP resource connected (CPUs, terminals, printers). A completed model of the real system must neither over simplify to the point where the model becomes trivial nor carry so much detail

that the model becomes clumsy and prohibitively expensive to operate (Ref 21:39). Emphasis will be placed on the predictive purpose of a system model.

Background and Current Resources

Prior to 1981, AFIT relied on its two major host organizations, the Air Force Logistics Command (AFLC) and the Aeronautical Systems Division (ASD) of the Air Force Systems Command for general purpose data processing support. However, the technological advances of the late 1970's and early 1980's, coupled with an increased demand for computational resources, forced AFIT to seek and develop internal data processing capabilities.

In 1981, the *PDP 11/34, PDP 11/60, and Harris* 500 were acquired and installed. They were almost immediately saturated, indicating the tremendous demand for computing resources. The VAX 11/780, acquired in October 1982, immediately relieved the capacity saturation problem for the above three resources. Since that time the PDPs have been taken off line for enhancements. The new machine has also quickly become saturated.

Computer Resources. Currently seven computer resources are available on the network. The CYBER actually consists of two computers - a CYBER 750 and CYBER 74. The CYBER 750 is primarily used for time-sharing (intercom) support while the CYBER 74 is used for batch processing. The use of these CYBER computers is transparent to users; therefore, the

HARRIS 500			HARRIS 80 RBT		
CPU Memory	1	(768 KB)	CPU Memory	1	(4 MB)
Disk Drives	2	(80 MB each)	Disk Drive	1	(40 MB)
Disk Drives	2	(300 MB each)	Tape Drives	2	
Tape Drives	4		Card Reader	1	
Card Reader	1		Card Punch	1	
Card Punch	1		Line Printer	1	
Line Printer	1		Pen Plotter	1	
			Printer/Plotter	1	

PDP 11/60			PDP 11/34		
CPU Memory	1	(256 KB)	CPU Memory	1	(256 KB)
Disk Drive	2	(28 MB each)	Disk Drive	1	(28 MB)
Printer/Plotter	1		Tape Drive	1	

VAX 11/780		
CPU Memory	1	(4 MB)
Disk Drives	2	(300 MB each)
Tape Drives	1	
Printer/Plotter	1	

Table 1.1 Hardware Description.

CYBER resource will be considered as one resource. The AFLC/CREATE system actually consists of two Honeywell H-635 processors operating in parallel. These also will be considered as one resource.

AFIT/ACD operates the Harris 500, PDP 11/34, PDP 11/60, Vax 11/780, and the Harris 80. Users to these systems consist entirely of AFIT people. The PDP 11/34 and PDP 11/60, which were obtained in 1981, have virtually been replaced by the VAX and are now only used by a few classes and faculty. It is anticipated that these systems will be taken off-line in 1984; therefore, they were not implemented in the network model. The Harris 80 functions as an input/output controller for the CYBER. Table 1.1 shows the hardware description of each AFIT network resource while

Computer	Computer ID	Interactive	Batch	# of Ports
CYBER	C	X	X	10
CREATE	R	X	X	16
Harris 500	H	X	X	28
VAX 11/780	V	X		38

Table 1.2 Computer Identification for Model.

Table 1.2 identifies the computers, interactive/batch capabilities and number of interactive ports.

Input Locations. The three schools of AFIT, Engineering, Logistics, and Civil Engineering are located in three separate buildings. Each school has both interactive and batch input locations. The interactive terminals are divided into two categories, student and administrative, while the batch locations are utilized by all users. The only exception is in building 125 where terminals exist for use only by software development personnel. The following input locations were defined:

1. STUD125 - Student terminals building 125
2. STUD640 - " " " 640
3. STUD641 - " " " 641
4. ADMN125 - Admin/Faculty terminals building 125
5. ADMN640 - " " " 640
6. ADMN641 - " " " 641
7. SOFT125 - Software Development building 125
8. BTCH125 - Batch input building 125
9. BTCH640 - " " " 640
10. BTCH641 - " " " 641

The location's attributes and interrelationships with the computers are shown in Table 1.3 The terminal input locations may access all of the computer resources. The batch input locations may access only a few of the computer

Input Location	Terminals Available	Computer Resource(s)	Hours of Operation
STUD125	5	ALL	0800-1600 Mon-Fri
STUD640	44	ALL	24 hrs daily
STUD641	14	ALL	24 hrs daily
FAAD125	1	ALL	24 hrs daily
FAAD640	5	ALL	24 hrs daily
FAAD641	6	ALL	24 hrs daily
SOFT125	6	ALL	24 hrs daily
BTCH125	-	R,H	0800-1600 Mon-Fri
BTCH640	-	C	24 hrs daily
BTCH641	-	R,H	24 hrs daily

Table 1.3 Network Input Locations.

resources. Dial-up was not considered as a separate input ocation. Each dial-up user is considered to be an arrival at one of the above locations (which the user would be if dial-up did not exist).

Output locations. Besides the terminals, seven hard-copy devices are available to the AFIT user. Some of these devices are restrained to one computer while others can be switched between computers. The Harris 80 print devices accept output only from the CYBER. The VAX 11/780 has one hard copy device. The Remote Batch Stations in buildings 125 and 641 can either accept output from the Harris 500 or the CREATE. The Harris 500 line printer accepts output only from itself. These interrelationships are shown in Table 1.4.

Methodology

The methodology applied in the research, is as Shannon describes, the art of modeling (Ref 21:19). It is an

Printer	Computer Resource(s)	Building Location	Hours of Operation
RBT125 Line Printer	R,H	125	0800-1600 Mon-Fri
RBT641 Line Printer	R,H	641	24 hrs daily
Har 80 Pen/Plotter	C	640	0730-2400 Mon-Fri
Har 80 Line Printer	C	640	0730-1600 Sat
Har 80 Print/Plotter	C	640	1200-2000 Sun
VAX Printer Plotter	V	640	
Har 500 Line Printer	H	641	0730-1800 Mon-Fri

Table 1.4 Network Printers.

intuitive art in which any set of rules for development of models can have limited usefulness at best and can only serve as a suggested framework or approach. However, we can use the experiences, judgement, and techniques employed by others.

Borovits and Neumann structure a simulation computer system model as three components. First, a workload model which is described in terms of job mix, rate of job arrival, and so forth. Next, a hardware model which defines the system components and their interrelations; the behavior of each component to different demands; and the decision rules for each component. Lastly, the simulation model incorporates the workload model and hardware model and quantifies the performance indexes (Ref 6:28). For this research, the three components Borovits described will be the workload model, network model, and system model respectfully.

No specific research found resembled the hardware configuration and complexity of the AFIT network. Basically, three areas were reviewed in order to employ the experiences, judgement, and techniques used by others. A

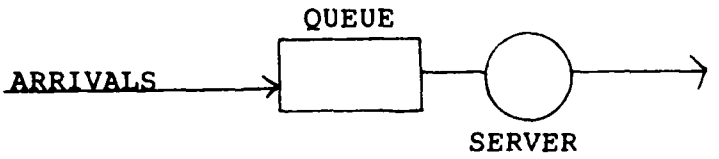


Figure 1.2 Single Server Queue.

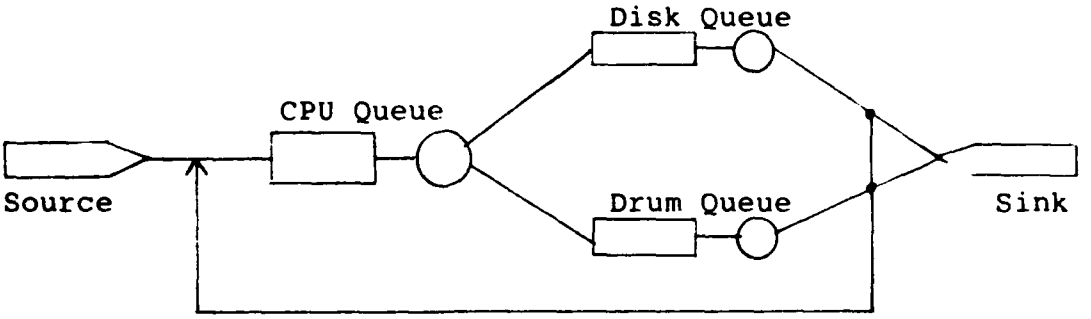


Figure 1.3 Open chain (batch load) (Ref 8:288).

review of analytical and simulation models is provided followed by a more comprehensive review of workload models.

Analytical Queueing Models. An analytical model is a set of equations that may describe the performance of a computer system. It is based on the fundamental mechanism in queueing theory as illustrated in Figure 1.2. It is not the purpose of this research to go into an in-depth coverage of queueing theory but Allen provides an excellent overview of queueing theory followed by applications to computer systems (Ref 3).

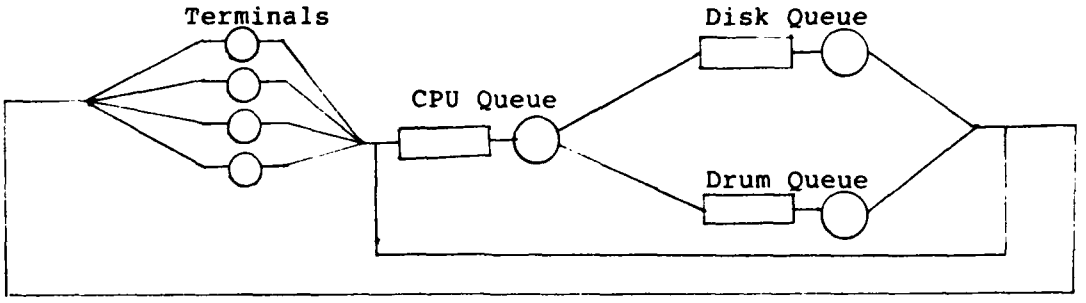


Figure 1.4 Closed Chain (terminal load) (Ref 8:288).

Chandy and Sauer illustrated computer systems by analytical queueing models (Ref 8:288). Figure 1.3 illustrates a computer system with an open chain representing a batch load. Figure 1.4 illustrates a computer system with a closed chain representing the terminal load. These systems can be combined to form a central server model with terminal and batch loads as shown in Figure 1.5. All of AFIT's computer resources can be represented by Chandy's and Sauer's combined terminal and batch load model. The VAX 11/780 does not have a card reader attached but interactive jobs may be entered in a batch mode. This is discussed in Chapter Three.

In a special issue on analytical queueing models in ACM Computing Surveys, Graham summarizes some of the basic reasons queueing network models have become so popular:

1) These models capture the most important features of actual systems, e.g. many independent devices with queues and jobs moving form one device to the next. Experience shows that performance

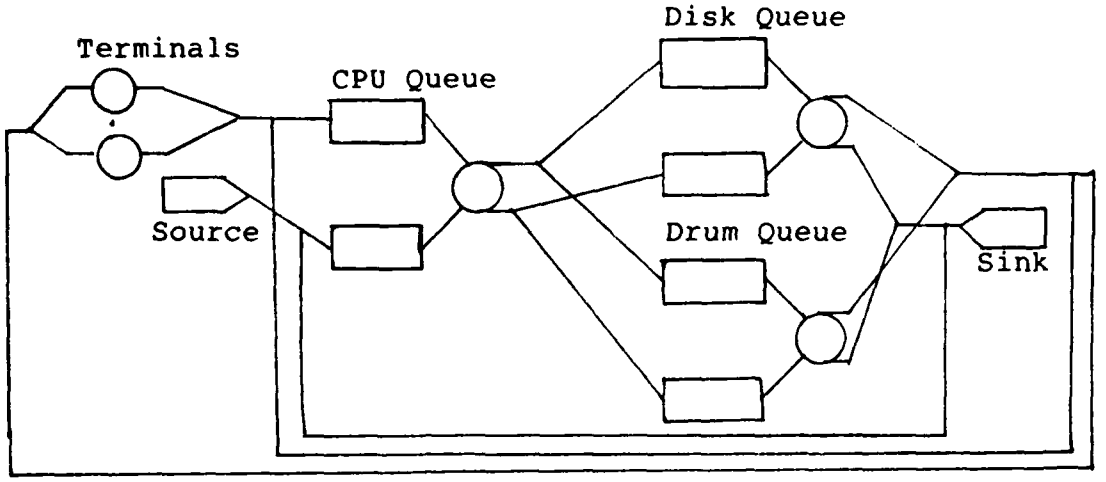


Figure 1.5 Central Server Model with Terminal and Batch Loads (Ref 8:289).

measures are much more sensitive to parameters such as mean service time per job at a device, than to many of the details of policies and mechanisms throughput the operating system (which are difficult to represent concisely).

2) The assumptions of the analysis are realistic. General service time distributions can be handled at many devices; load dependent devices can be modeled; multiple classes of jobs can be accommodated.
3) The algorithms that solve the equations of the model are available as highly efficient queueing network evaluation packages. (Ref 12:220)

In another special issue on analytical queueing models in Computer, Spragins continues:

> Another very important reason for the increasing popularity of these models is simple: they work. In fact, they have been found to be surprisingly successful in estimating such performance metrics as throughputs, mean queue lengths, and mean response times for real systems. The amazing thing is that they have been successful despite the fact that a number of the most common assumptions, such as time invariant parameters, steady state operations, and Markovian - or similar - behavior, are often

> seriously violated by real systems. That relatively simple models based on demonstrably false assumptions still give good results is encouraging to the practitioner but puzzling to the analyst. (Ref 22:9)

Simulation Models. Computer simulation models are employed whenever a system is at least partially known but it is too complex to handle by means of an analytical model (Ref 6:26, Ref 16:138). The models are still formulated using queueing theory as described earlier, however an analytical model becomes computationaly unwieldy and many times requires simplifying assumptions. Analytical models also require more knowledge of the system than may be available.

Simulation models may apply to entire systems or subsystems and components at any level of detail. Computer systems are usually modeled as discrete systems which are characterized by discrete states and by state transitions (events) occurring at discrete time instants (Ref 11:102). A simulation moves the system through time and mimics the myriad events that occur in the real world.

Computer simulation models are often used in a computer performance evaluation effort (Ref 6,11,15,16). Commercial simulation program packages or 'in-house' simulations may be used. Commercial simulation packages are useful in solving routine problems such as configuration selection where a high level of detail is not involved. The configuration in most cases is limited to a single computer and its peripherals which may be added. Usually, simulations of complex configurations are not highly reliable.

Additionally, a user is not always aware of the details, approximations, and simplifications built into ready made packages. In-house simulation models of complex systems are usually more reliable and understandable. The in-house model will also be more efficient since it embodies only the components and functions relevant to the user. Of course, in-house simulations are usually more costly to operate (Ref 6:33-34).

Usually, most computer simulation modeling efforts are directed towards specific computer systems. Nguyen et all describes a detailed simulator which predicts the effects of change in IBM 3790 and 8100 distributed processing systems and teleprocessing networks (Ref 18:81). This simulator, unformally called FIVE, relies heavily upon IBM developed monitoring aids to drive the simulation. Markowitz outlines a IBM 370 simulator written in SIMCRIPT II which was designed to model the virtual memory of the IBM 370 series (Ref 15:342).

None of the network simulation efforts viewed resembled the AFIT network. The efforts viewed either simulated specific systems or if simulated networks, the access protocols, channel messages throughput, and delays were simulated (Ref 9:123). Because the AFIT network consists of four different vendor's computers, specific system simulations can not be utilized. The prediction of network protocols, channel messages throughput, and delays is not an objective of this research.

Workload Models. Generally, workload models serve four purposes. A representative workload is provided for comparative performance evaluation of different systems. For experimental performance optimization studies, they provide a controllable reproducible environment. A workload model may also reduce the quantity of data that has to be analyzed. Finally, they present the system workload in a form required by a system model (Ref 23:52).

For this research, the primary purpose of the workload model is to drive the network model. A computer system (in this case network) may be treated as a passive device which carries out data processing tasks in response to requests made by its users (Ref 2:19). Since each computer system (network) consists of a number of hardware and software resources, a user request requires a certain amount of these resources in order to be satisfied.

Workload has traditionally been defined in terms of a set of job parameters representing resource demands such as CPU time, I/O time and central memory (Ref 23:10). Defining these terms for a model has become known as the workload characterization problem. Many times this characterization becomes the hardest technical problem to solve (Ref 11:221). Characterizing a workload for evaluation purposes requires determining which of the many parameters have an influence on the system's performance.

Agrawala (Ref 2:19) characterized the workload by defining a user request R into a vector X where the

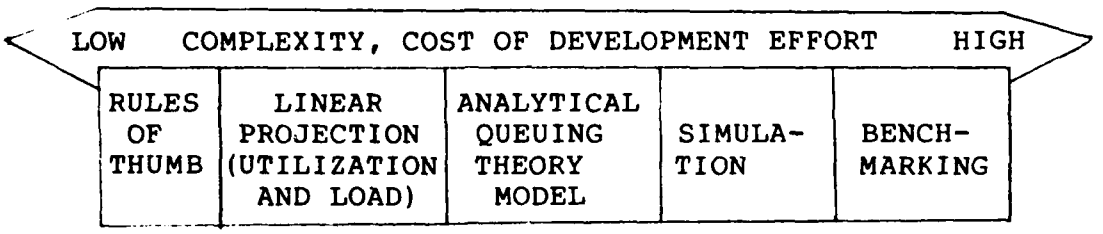


Figure 1.6 Spectrum of Computing System Modeling Techniques (Ref 3:13).

components of X represent the amount of service requested from the resources to service the requests. For example, X(1) may be the number of memory blocks and X(2) the CPU time required.

Agrawala further defines the request with other characteristics. It is also made up of a time instant T and from a location L. T may be the user arrival or job arrival. L would be the input location. It also may have a flag F associated with it which indicates whether it is a time sharing, a batch, or a real-time processing request. Therefore, any user request may be characterized by the quadruple (T,L,X,F) which may be considered to be a multivariate point process over time and space (Ref 2:19).

Applied Methodology. This section provides a general description of how modeling methodology was applied in this research. Figure 1.6 illustrates the spectrum of computer system modeling techniques. The expense in time and effort runs from the least, rules of thumb, to the costliest, benchmarking. Rules of thumb are based on the experiences

of managers. It works well for short periods (initial computer acquisition), but it declines in value over time and is not much help in predicting hardware and software upgrading requirements (Ref 3:13, Ref 10:19). Linear projection can be a helpful planning tool but the major problem is that this approach uses linear projection for a system that is inherently nonlinear. Analytical queueing models have been shown to work well (previous section) provided that the interaction of hardware performance factors and software are understood. Simulation models are used when equations are insoluble or when it is not obvious what the equations are. The system can be modeled at much greater detail than is practical in queueing theory models. Lastly, benchmarks involve taking an entire workload or a partial workload from one CPU and putting it on another. The results are usually more precise and real than the other methods but it is time consuming and a nuisance (Ref 10:81).

For this research the above techniques except benchmarking are combined. The methodology becomes a hybrid simulation that uses analytical models to replace subsystems in the simulation of the entire system, thereby simplifying the overall simulation (Ref 12:220). For example, within a computer, the I/O system may be considered a subsystem and its performance quantities are calculated from an analytical model. Thus, the simulation does not need to be concerned with the low level detail of the I/O subsystem. Linear projection may be used in some cases where a better solution

is not available or a complete understanding does not exist.

Shannon distinguished eleven stages that may be used to investigate the properties of a real system as follows:

1. System Definition--Determining the boundaries, restrictions and measure of effectiveness to be used in defining the system to be studied.
2. Model Formulation--Reduction or abstraction of the real system to a logic flow diagram.
3. Data Preparation--Identification of the data needed by the model, and their reduction to an appropriate form.
4. Model Translation--Description of the model in a language acceptable to the computer used.
5. Validation--Increasing to an acceptable level the confidence that an inference drawn form the model about the real system will be correct.
6. Strategic Planning--Design of an experiment that will yield the desired information.
7. Tactical Planning--Determination of how each of the test runs specified in the experimental design is to be executed.
8. Experimentation--Execution of the simulation to generate the desired data and to perform sensitivity analysis.
9. Interpretation--Drawing inferences form the data generated by the simulation.
10. Implementation--Putting the model and/or results to use.
11. Documentation--Recording the project activities and results as well as documenting the model and its use. (Ref 21:23)

This research will pursue stages 1 through 5. Further stages are not possible because data needed by the model were not available or in appropriate form for experimentation. This will be addressed in more detail later. Documentation of the model and its use were also completed. This stage is an on going stage which should not be placed last.

A precise methodology cannot be stated. Shannon's stages above may provide 'an order of attack' but the fact

remains that modeling can best be described as an intuitive art (Ref 21:19). As Shannon states:

> Thus, the art of modeling consists in an ability to analyze a problem, abstract from it its essential features, select and modify basic assumptions that characterize the system, and then enrich and elaborate the model until a useful approximation results. (Ref 21:20)
>
> ...The art of modeling can be mastered by those who possess the necessary skills of ingenuity, insight, and resourcefulness, as well as an extensive exposure to systems and physical phenomena they are trying to model. There is no hard and fast rule about how the problem is originally formulated, i.e. how one looks at it in the first place. There are no magic formulas for deciding what should be included in the model in the form of variables and parameters, descriptive relationships and constraints, and criterion for judgement of effectiveness. Remember that nobody solves the problem; rather, everybody solves the model that he has constructed of the problem. This concept helps to keep the model and art of modeling in the proper perspective. (Ref 21:21)

In summary, Shannon provides a framework (11 stages) that a modeler may use to develop a model. However, the modeler must relay upon the skills of ingenuity, insight, and resourcefulness to successfully complete a model. During the system definition and model formulation stages the modeler must decide what are the boundaries and restrictions, variables, and parameters. The final judgement of the model's worth will be determined by the users.

Specific Approach

Starting with Borovits' and Neumann's description of a computer simulation model, the entire system model can be broken into two smaller components (models) -- the workload component and the network component. Each of these

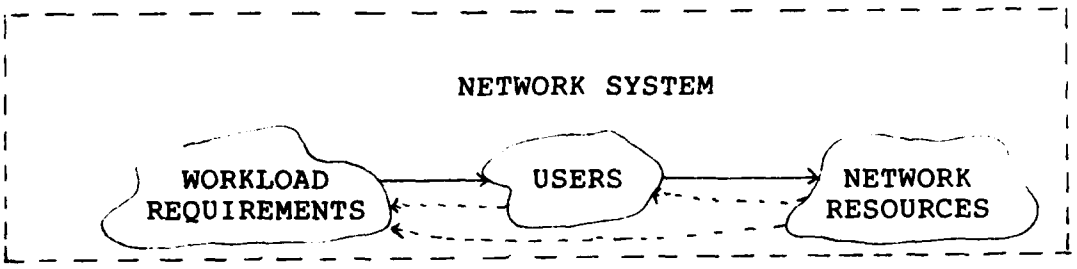


Figure 1.7 Basic Network Conceptualization.

components can be modeled and evaluated individually before combining them to form the system model. This section provides the overall conceptualization of the AFIT network which is divided into the workload and network components. The interface between the components is defined by modifying Agrawala's vector to fit the situation at AFIT. Lastly, a simulation language used by this research is described.

Basically, the network system was seen as workload requirements which cause users (arrivals) to request and utilize the network resources as illustrated in Figure 1.7. The dashed lines represent the constant feedback which may influence the user arrivals and workload requirements. For example, poor response time from the network resources will irritate the users which then may change the workload requirement. When possible all distributions used by the model are tied to current literature. Basic subsystems (CPUs) were modeled using analytical models or linear projection when possible (an understanding exists).

The workload characterization problem will be handled by modifying Argrawala's vector as described in the previous section. Agrawala defined his model as R=f(T,L,X,F) where:

R = Request
T = Time of the request
L = Location of the request
X = Resource vector
F = Timesharing/Batch Flag

The F flag may be dropped because the timesharing/batch option becomes a function of location L. The input locations of the model determine whether it is an interactive or batch request. The time of request may be represented as an arrival rate for each input location (node) of the network. The network model may then determine simulated arrival times for each location based upon this empirical arrival rate. O'Neil and O'Neil (Ref 19:11) found the pattern of interactive arrivals best treated by an empirical distribution.

We are left with the definition of the vector X. Besides the traditional definition of X which includes CPU time, core memory, and so forth, it is necessary to identify for the network model the computer resource required. With CPU time, memory blocks, and so forth dependent upon each computer system an attempt to describe the job parameters as a probabilistic description would be extremely difficult to manage (Ref 13).

Agrawala and later Hartrum and Thompson simplified the description by use of 'cluster analysis' (Ref 13). By use of cluster analysis a large, complex distribution is divided into a number of simplier distributions. Using this technique a workload can be characterized into an empirical

distribution of these clusters. Keeping in mind that the model is being developed for management for prediction purposes, these can later be identified by data analysis. These clusters should have meaning to management people without a technical background. For example, a type of work can be identified such as basic language development and then within that type the clusters can be identified. Using this technique, the workload model is independent of the hardware performance in the network model. After an arrival at a input node of the network the following resource requirements will have to be identified by the network model:

1. Computer required.
2. CPU time required.
3. I/O time required.
4. Memory required.
5. Printer required.

These resource requirements can be determined by providing an empirical frequency distribution. The workload model can provide these resource requirements through a cumulative probability distribution by computer, job class, and job size. The computer should not only include specific identification, but also a probability that 'any' computer will satisfy the user request. The job class will identify the 'type' of work being performed such as basic language development or simulation. From within these 'type' of classes, the job size (clusters) can be used by the network model to determine CPU time, I/O time, and memory required.

The printer selection becomes a function of the type work, location, and computer selected. The network model will be able to determine this from empirical data. Hence, the workload can be broken up into n computers, m classes and k sizes.

The simulation language used in this research is SLAM which is an advanced FORTRAN based language. It allows simulation models to be built with three different world views--network, discrete, and continuous (Ref 20). It lends itself very well to Shannon's discrete change simulation as follows:

1. The world is viewed as a set of entities that may be modified or qualified by their characteristics, call attributes.
2. The entities interact with specific activities of the world consistent with certain conditions, which determine the sequence of interactions.
3. These interactions are regarded as events in the system, which result in changes to the state of the system. (Ref 21:108)

Within AFIT's network 'world' the user arrivals and computer jobs become the entities. Activities of the world would include a user attempting to log on or a batch job entered into a card reader. CPU arrivals would be an event.

Order of Presentation

Discussed in chapters two through four are the implementation of the research methodology and recommendations for future research.

Described in Chapter Two, The Workload Model, are the workload model and interface to the network model. The users of the network were identified and categorized. The implementation of the resource vector X as described in this chapter is described and the arrival rate calculations are defined. Behavioral and outside influences to the workload are discussed. Lastly, the model is presented.

Chapter Three, The Network Simulation Model, describes the network model, operation of the model, and how the model reflects the AFIT network. The conceptualization of the network system is presented followed by a more detailed look at the network resources. The variables of the model are discussed in detail.

Presented in Chapter Four, Summary, Recommendations, and Conclusion, are the model summary and recommendations for future research. A capacity planning methodology is reviewed.

Chapter 2

THE WORKLOAD MODEL

Introduction

This chapter describes the AFIT workload, the workload model, model operation, and interfaces to the network model. First, users of the network were identified and categorized. Next, the resource vector X (as described in the previous chapter) was defined and the arrival rate calculations were determined. The behavioral aspects and other outside influences to the workload were also considered and some were incorporated into the model. Finally, the workload model was developed to transform the identified workload characterization into the form required to drive the network model.

The Users

The workload imposed at AFIT, like most universities, is unique and not typical of the commercial environments (Ref 14:304). Iyengar and Chih-Chun grouped the Louisiana State University network users into the following ten groups (Ref 14:306):

1. group 0 - administrative related jobs
2. group 1 - classwork for student users in lower level courses
3. group 2 - classwork for student users in senior or graduate level courses
4. group 3 - testing and grading departments classwork
5. group 4 - faculty preparation of classwork
6. group 5 - masters level research
7. group 6 - doctoral level research
8. group 7 - faculty research funded by the department
9. group 8 - research funded by a grant or contract
10. group 9 - governmental agencies or commercial application

CATEGORY NUMBER	FUNCTIONAL SUPPORT
A	Academic/Educational Support
A-1	Student Course Work
A-2	Classroom Exercises
A-3	Library Services
A-4	Laboratory Support
A-5	Special Laboratory/Department Support
A-6	Teaching Support
A-7	Test Preparation and Grading
A-8	Course/Class Critiques
B	Research and Professional Development
B-1	Student Thesis Research
B-2	Sponsored Faculty Research
B-3	Internal Faculty Research
C	Registrar Support
C-1	Student Selection and Admissions
C-2	Registration and Scheduling
C-3	Educational Plans and Transcripts
C-4	Professional Continuing Education
D	Civilian Institution Support
D-1	Student Information/Education Plans
D-2	Special Programs
D-3	Budgeting and Accounting
E	Management and Administration
E-1	Budgeting and Accounting
E-2	Plans and Programs
E-3	Personnel Records/Data
E-4	Recurring Reports
E-5	Special Reports

Table 2.1 ACD Functional Support Categories (Ref 4:4).

AFIT ACD defines and classifies its support into 5 functional categories (Ref 4:4) as shown in Table 2.1.

Both Iyengar/Chih-Chun and ACD's categories can be combined and re-grouped. Basically these categorizations support four types of users--students, faculty, admini-

stration, and software support and development. The demands placed by students are course work, word processing, and thesis work. The faculty's demands are administrative (word processing, classroom preparation) and research. The administrative demands can be divided into daily requirements (word processing), weekly requirements (weekly reports), and periodic requirement (scheduling). Finally, at AFIT a software development shop exists. These categories are summarized below.

1. Student
   - Student course work
   - Student word processing
   - Student thesis work
2. Faculty
   - Faculty course preparation
   - Faculty research
   - Faculty word processing
3. Administration
   - Daily requirements
   - Weekly requirements
   - Periodic requirements
4. Software Development

From these categories the input files to the model can be defined as discussed later.

Resource Vector X

Each user has his choice of up to four different computers and, depending on the computer selected, up to three different hard copy output devices. The user may use direct connect terminals at AFIT locations, dialup terminals available for checkout, or personally owned terminals/ computers. Batch may be utilized on all computers except the

VAX 11/780. Therefore, for each user/requirement, whether it be interactive or batch, a resource vector X is identified as follows:

Computer Required
Class Work
Size Work

The computer required will identify for the network the specified computer required or 'any' computer that will satisfy the requirement. If the 'any' computer is identified, the network model will have to determine the computer resource as described in the next chapter. The class work and size work identify for the network model the 'clusters' as described in the previous chapter. The network model will transform these cluster definitions into the actual parameters which are CPU time, I/O time, and memory requirements. For this model's development, the clusters and subsequent parameters are derived rather than empirically defined.

Software or hardware constraints may dictate which computer will be used. It also may be dictated by an instructor. In some cases user requirements can be satisfied by two or more of the computer systems. The Computers were defined by using the following identification codes:

COMPID	Description
C	CYBER
R	CREATE
H	Harris 500
V	VAX 11/780
A	Any

As described earlier, the CPU time, I/O time, and memory requirements can be expressed as a function of the class of work and size within that class. Table 2.2 shows the software available on each computer system. From this Table the workload classes were defined as follows:

CLASID	Description
A	Basic Languages
B	DBMS
C	SPSS (mathematical)
D	SLAM (simulation)
E	Canned Routines
F	Word Processing

These classes should be broadly defined in order that management may conceptualize the requirements.

Within each class five memory sizes (clusters) were defined as follows:

SIZEID	Description
1	< 20K
2	20K - 40K
3	40K - 60K
4	60k - 100K
5	> 100K

The memory size descriptions would be determined by use of cluster analysis as outlined in the previous chapter. The descriptions used here are hypothetical. For this model's development the description refers to the memory size distribution. The distributions for memory, CPU seconds, and I/O seconds for each cluster would be input to the network model as is discussed in the next chapter. The data to determine these clusters does not exist.

SOFTWARE	CYBER	CREATE	HARRIS 500	VAX11/780
1. Languages				
-Basic	X	X	X	X
-COBAL	X	X	X	
-FORTRAN	X	X	X	X
-PASCAL	X		X	X
-ALGOL		X		
-JOVIAL		X		
-SNOBOL	X		X	
-RATFOR	X			X
-APL				X
2. Data Base Mgt Systems	VENUS	IDS	TOTAL	INGRES
3. Math, Statistics and Simulation				
-SPSS	X	X	X	
-SCSS		X		
-SLAMII	X			X
-QGERT	X	X	X	
-DYNAMO	X			
-TOTAL	X			
-MPOS	X			
-GASP	X	X		
-MIMO	X			

Table 2.2 AFIT Network Software.

After the expected number of sessions (arrivals) are tabulated for a input location by computer, class, and size a cumulative frequency distribution can be calculated as follows:

```
COUNT=0.0
FOR I=1,NUMCPU
     FOR J=1,NUMCLS
          FOR K=1,NUMSIZE
               OBSFRQ(K)=FREQ(I,J,K)/TOTSES
               COUNT=OBSFRQ(K)+COUNT
               FREQ(I,J,K)=COUNT
          ENDFOR
     ENDFOR
ENDFOR
```

where;

NUMCPU = Number of computers.
NUMCLS = Number of work classes.
NUMSIZ = Number of sizes.
FREQ = Array initially holding session count by computer, class and size.
TOTSES = Total number of sessions for week.
COUNT = Cumulative Frequency.

This simple algorithm creates a cumulative frequency distribution which defines the resource vector X. The total size of the distribution is the number of computers (NUMCPU) times the number of classes (NUMCLS) times the number of sizes (NUMSIZ). The total number of sessions (TOTSES) will need to calculated beforehand. The observed frequency for each computer, class, and size is calculated by dividing the number of sessions for each computer, class, and size by the total number of sessions. COUNT is used to keep the cumulative frequency. Finally, this cumulative frequency is stored in the FREQ array replacing the number of sessions.

This cumulative frequency distribution is output by the workload model for use by the network model. The network

model will determine the job's parameters by generating a random number which will correspond to a specific computer, class, and size in this distribution. From that description, the network model will determine a job's CPU time, I/O time, and memory requirements. This transformation will be described in the next chapter.

Arrivals

For all interactive requirements a certain number of interactive sessions would be required to complete the work. Each work class has a certain complexity and this complexity increases as the size increases. Therefore, the number of sessions required to complete was hypothesized to be a function of the work class and size. Likewise for batch, the number of batch runs reflects the same hypothesis.

Given the total weekly requirements by class of work, size of work, and location, the total number of arrivals for each location needs to be distributed over the week. O'Neil and O'Neil (Ref 19:11-12) concluded that the distribution of arrivals within a day could best be treated as an empirical bimodal distribution and that the groups to which users belonged is a significant factor. Hence, it is hypothesized that the distribution of interactive and batch arrivals across the week would differ. Furthermore, within the interactive users, students and faculty/administrative distributions would differ. The arrivals at the software development shop is hypothesized to follow the administrative distribution.

From the daily rates it would be necessary to distribute the arrivals on a hourly basis. The same hypotheses concerning weekly distributions apply over the day.

After the total expected arrivals (interactive and batch jobs) are tabulated for a week, the arrivals can then be distributed over the week for each location as follows:

```
FOR  I=1,7
     DAYRATE(I)=TOTSES*DAYDIST(I)
     FOR  J=1,24
          HOURRATE(I,J)=DAYRATE(I)*HOURDIST(J)
     ENDFOR
ENDFOR
```

where;

TOTSES = Total expected sessions for the week
DAYDIST = Distribution of expected workload over week.
HOURDIST = Distribution of daily workload over day.
DAYRATE = Daily arrival rate.
HOURRATE = Hourly arrival rate.

First, the total number of sessions (TOTSES) for each week is distributed across the days of the week. This is accomplished by multiplying the total number of sessions by a daily frequency distribution (DAYDIST). The results are stored in DAYRATE. The hourly rate can then be calculated by multiplying each day's rate (DAYRATE) by a hourly frequency distribution (HOURDIST). Finally, hourly arrival rates for an entire week (HOURRATE) is output for latter use by the network model.

Behavioral and Outside Influences

Like any system a computer system is effected by user behavior and outside factors. Besides the arrival rates identified above, the user has a option of using interactive

or batch input mediums. For example, oldtimers have used and are comfortable with cards while new users may not have ever seen cards. The input medium can be dictated by the instructor. Novice users may be only provided batch procedures. The service provided may differ between interactive and batch; therefore, the user may feel it is easier to use some mediums over others.

Another outside factor has been the increasing influx of personal computers (PCs). These computers can take off some of the requirements placed on the system resources whether as a stand alone device or as an interactive terminal accessing the network. The number of PCs bought would also differ from the type of user. Engineering students and faculty probably have more than Logistics or Civil Engineering personnel. Coupled with the decision for the user to purchase a PC is the perceived service provided by the network.

Lastly, the resources used by a user is influenced by 1) first system taught, 2) perception of services and 3) ease of use. For example, some users will consistently use one resource because they are familiar with it regardless of difficulty of use. Others will try all until they find their perception of the easiest or most efficient. Finally, some will purposely choose the 'difficult' system because of the lack of use by others and therefore better turnaround time.

The behavioral and outside influences discussed above are only a few of the many outside factors which will affect the workload and subsequently the network. All of the behavioral and outside influences would be extremely difficult to identify and understand much less include in the model. Therefore, only two of the most important factors where included in the workload model. First, the influx of personal computers was included because their use directly satisfy requirements which would be placed upon the network resources. Second, the choice of interactive or batch mediums is included. This choice will directly affect the number of terminals required on the network and the overhead of each computer as more interactive users are attached.

The Model

The primary purpose of the workload model is to provide the values needed to drive the network model over the period of a school quarter (11 weeks). This output consists of the arrival rates and cumulative frequency distribution of resoures required for each input location of the network as defined in Chapter One. Both of these are empirical distributions. Though ideally it would be best to verify and validate each type of user or location separately, this is not possible because of the interconnection (batch) of both.

Inputs. The categories of users defined earlier were transformed into separate input files to define each category's workload requirements. These files were struc-

tured to define the workload requirements at AFIT for an entire year (four school quarters).

The student load was defined into three input categories: course work, word processing, and thesis work. Course work was identified by location, course, number enrolled, quarter offered, the work required by week (computer, class, size), and the behavioral factors of PC and BATCH. PC is the expected percentage of students which own personal computers while BATCH is the expected percentage of students which will use batch if possible. Word processing requirements are defined for each school. Within each school the number of students, percent using network word processing, and a load factor per week are defined. The load factor is defined as percentage of the probable students expected to use word processing per week over the quarter. The student thesis work was structured like the course work.

The faculty load was defined by two input files -- faculty research and word processing. The faculty research and word processing files are structured like the student files.

Administrative load was defined by three files -- daily requirements, weekly requirements and periodic requirements. Daily requirements define the expected daily load and was structured similar to student course work. Weekly requirements and periodic requirements identify specific requirements by day, time, computer, class, and size.

Input File	Format	Description
CRSWRK	1	Student course work
THSWRK	1	Student thesis requirements
STUDWP	2	Student word processing
FACRCH	1	Faculty research and course preparation
FACTWP	2	Faculty word processing
DAYREQ	3	Daily administrative requirements
WEKREQ	4	Weekly administrative requirements
PERREQ	4	Periodic administrative requirements
SFTDEV	3	Software Development requirements

Format	Description
1	Location, Course/Description, quarters, # of students/users, work per week (computer,class, size), PC factor, BATCH factor
2	Location, quarter, # of users, percent, load per week, PC factor
3	Location, description, quarter, computer, class, size, # users, BATCH factor
4	Location, description, quarter, computer, class, size, day number.

Table 2.3 Workload Input Files.

Software development requirements are defined exactly like the daily requirements file above. Table 2.3 shows the names and structure of these files.

Table 2.4 identifies the input variables required to drive the workload model. BAFACT identifies the computers which have batch capability. All except the VAX 11/780 have batch capability. BTCHFACT array defines the expected number of batch runs required by class and size. DAYSAD, DAYSBA, and DAYSST arrays are the administrative, batch and

Variable Name	Description
BAFACT(MAXCPU)	Identifies whether computer has batch capability.
BTCHFACT(MAXCLS, MAXSIZ)	Expected number of batch runs required by class and size.
DAYSAD(7)	Distribution of administration workload over the week.
DAYSBA(7)	Distribution of batch workload over the week.
DAYSST(7)	Distribution of student workload over the week.
HOURAD(24)	Distribution of administration workload over a day.
HOURBA(24)	Distribution of batch workload over a day.
HOURST(24)	Distribution of student workload over a day.
SESFACT(MAXCLS, MAXSIZ)	Expected number of sessions by class and size.
PCFACT(MAXCLS, MAXSIZ)	Identifies classes and sizes able to process on PC.
WPFACT(MAXCPU, MAXSIZ)	Word processing distribution over computers by size.

Table 2.4 Main Workload Input Variables.

student distributions of the workload over the week. HOURAD, HOURBA and HOURST arrays are the administrative, batch and student distributions of a day's workload over the day. PCFACT array identifies the classes and sizes capable of being run on a personal computer. For example, FORTRAN and Pascal jobs would be expected to run on a PC while SLAM and SPSS would not. SESFACT array defines the expected

number of sessions needed to complete a requirement by class and size. WPFACT array distributes by size the word processing requirements over the available computers. All of these values would have to be determined from data analysis after job sizes (clusters) are defined.

Student Locations. Each file was read and distributed into three arrays; Total Interactive (TOTINT), Total Personal Computing (TOTPC), and Total Batch (TOTBAT). The number of users were calculated for each physical location (LOCAT), week (WEEKNO), computer (COMP), class (CLASS), and size (SIZE). For student course work the distribution was as follows:

```
TOTPC(LOCAT,WEEKNO,COMP,CLASS,SIZE) =
                NUMSTUDS * PCFACT(CLASS,SIZE) * INPC
TOTBAT(LOCAT,WEEKNO,COMP,CLASS,SIZE) =
                NUMSTUDS * BAFACT(COMP) * INBTCH
TOTINT(LOCAT,WEEKNO,COMP,CLASS,SIZE) =
                NUMSTUDS - (TOTPC(...) + TOTBAT(...))
```

where;

```
NUMSTUD = number of students requiring resources COMP,
          CLASS and SIZE at LOCAT
INPC = input PC factor
INBTCH = input BATCH factor
```

The total number of PC users (TOTPC) is equal to the total number of students (NUMSTUD) times a PCFACT (as described earlier) times the input PC factor (INPC). The input PC factor is the expected percentage of NUMSTUD which own personal computers. The total number of batch users (TOTBAT) is calculated in the same manner using BAFACT and INBTCH factors. BAFACT defines whether this computer has

batch capabilities and INBTCH is the expected percentage of NUMSTUD which would use batch instead of interactive mediums. Lastly, the total number of interactive users (TOTINT) is equal to the total number of students minus the PC users and the batch users.

Thesis work was distributed the same way. Word processing work was determined and distributed as follows:

```
NUMWP(WEEKNO) = NUMSTUD * PERCENT * LDFACT(WEEKNO)
```

where;

```
NUMWP = number word processing users
NUMSTUD = input number of students
PERCENT = expected percentage of students to use word
          processing
LDFACT = expected load/week
```

The number of word processing users (NUMWP) for each week is calculated by multiplying the total number of students (NUMSTUD), the expected percentage of students to use word processing (PERCENT), and the load factor per week (LDFACT) as described earlier. These users were then distributed into the 'total' arrays as follows:

```
TOTPC(LOCATION,WEEKNO,COMP,CLASS,SIZE) =
    NUMWP(WEEKNO) * WPFACT(COMP,SIZE) * INPC
TOTINT(LOCATION,WEEKNO,COMP,CLASS,SIZE) =
    NUMWP - TOTPC(...) * WPFACT(COMP,SIZE)
```

The total number of PC users is equal to the total number of word processing users (NUMWP) times the word processing factor (WPFACT) and the input PC factor. The WPFACT array defines the distribution of word processing requirements for each computer. The input PC factor is again the expected

percentage of NUMWP who own personal computers. The total number of interactive users (TOTINT) is equal to NUMWP minus the personal computer users times the same WPFACT.

The arrival rates and cumulative frequency distributions are then calculated for the three student locations (STUD125, STUD640, STUD641) from TOTINT by multiplying TOTINT by SESFACT. The arrival rates and cumulative frequency distributions are written to an unformatted output file for the student locations.

Faculty/Administrative Locations. The faculty and administrative categories are combined since they use the same input locations. The TOTINT and TOTPC arrays are cleared and the faculty research and word processing requirements are processed like the student's. The daily administrative file is also processed like the student course/thesis and faculty research except there is not a PC factor. The arrival rates and cumulative frequency distributions are then calculated for the three faculty/administrative locations (FAAD125, FAAD640, FAAD641) from TOTINT. The number of sessions and output files are calculated and written exactly like the student locations.

Software Development Location. The TOTINT array is again cleared and the software development file is processed. The file is structured and processed like the daily administrative requirements file. The output arrival rates and cumulative frequency distributions for location SOFT125 is calculated and written.

Batch Locations. After all the above locations have been processed, the TOTBAT array contains all of the batch requirements. The three batch location's (BTCH125, BTCH640, BTCH641) arrival rates and cumulative frequency distributions can be figured and written using BTCHFACT instead of SESFACT.

Constant files processing. The input files weekly and periodic requirements contain resource requirements which cannot be included into the arrival rates and frequency distributions. Each record in the weekly file is read and eleven records are created. These eleven records correspond to the eleven weeks (ten plus finals) in the quarter. The periodic file can then be read and merged into the output file.

Verification and Validation

Verification of the model required preparation of the input files and selected print statements within the modules. The workload requirement input files were created to test all paths within the modules. These files were kept small enough to verify manually the expected output of the model. The input variable values were chosen to represent reality although few of these values can be validated.

The test output of the workload model matched the expected output calculated manually beforehand. The print statements inserted in the modules verified the paths and transformations made by the modules. The output file format was verified after completion of the network model.

Chapter Summary

Chapter Two has presented the workload conceptualization and characterization of the workload necessary to drive the network model. The interface between the workload and network models has been established. Chapter Three presents the network model.

Chapter 3

THE NETWORK SIMULATION MODEL

Introduction

This chapter describes the network simulation model, model operation, and how the model reflects the network. The network simulation model was developed in three phases. First, a conceptual structure of the primary components as described in Chapter One was created. Next, the input variables were identified. Besides the arrival rates and cumulative frequency distributions created by the workload model, tables and probability mass functions were created to describe the complex interactions of the components and complex routing decisions. Next, the conceptual structure was translated into the simulation language. Finally, verification and validation of the model is discussed.

Conceptual Structure

The AFIT network system consists of three types of resources--input locations, computers, and printers. These resources are depicted in Figure 3.1. The input locations are then divided into terminals and batch (card readers) locations. Some of the batch locations may be switched between computers. Each computer has a maximum number of input ports which limit the number of terminals connected. For example, the VAX 11/780 has 38 ports connected; therefore, only 38 interactive users can be connected. Some of the printers may be switched between computers as described in Chapter One. Their selection is always

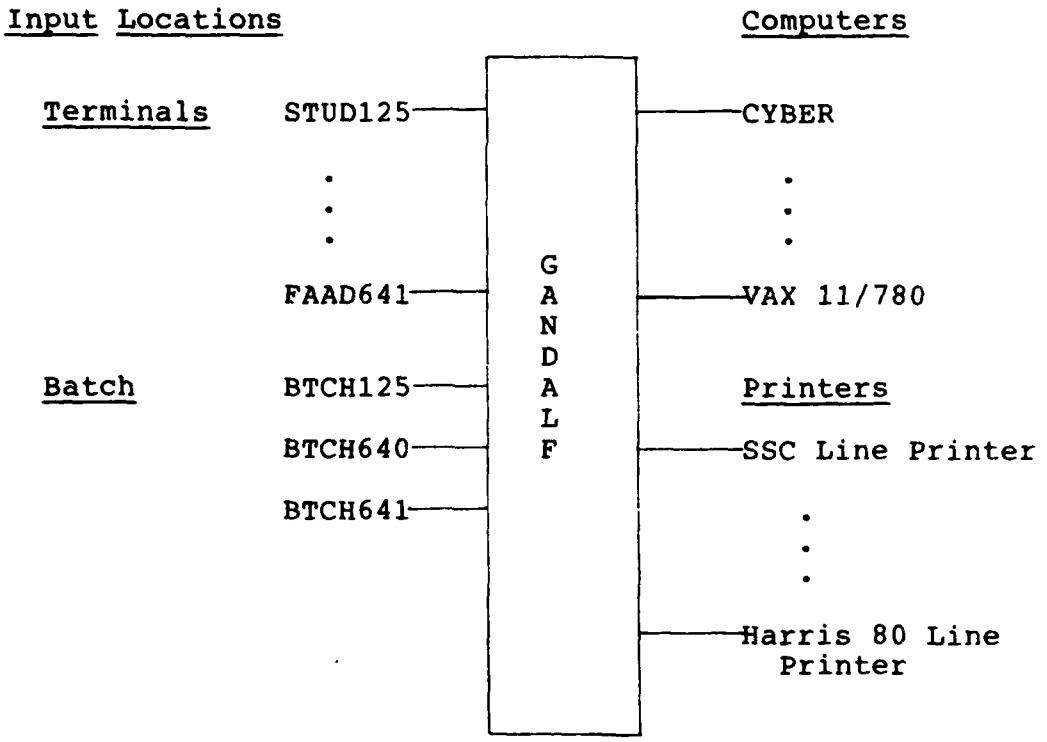


Figure 3.1 Overall Network Conceptualization.

dependent upon the computer selected and input location.

The GANDALF switchs between the resoures is not simulated. The prediction of network protocols, channel message throughput, and delays are not an objective of this research. Although the capability of intercomputer communications is possible at AFIT, the software is not available. It is not expected to be required in the future except for the occasional transfer of files.

Arrivals. There are two types of arrivals to the network, interactive users and batch jobs. After the interactive user arrives, a series of processes must be accom-

Condition	Response
Computer Not Available	Leave - Reschedule Arrival
Computer Available but Terminal Not Available	Wait for Terminal or Leave - Reschedule Arrival
Computer and Terminal Available but Port Not Available	Wait for Port or Leave - Reschedule Arrival
Computer, Terminal, and Port Available	Log-on

Table 3.1 Interactive Arrival Responses.

plished before connection to a computer is established. These conditions and responses are summarized in Table 3.1. The user must determine if the computer, terminal (if not dial-up), and port into the computer are available. If the computer is not available, the user leaves and returns later. If the computer is available, a terminal is then seized. When no terminals are available, the user must decide whether to wait or to return later. Even after the computer and terminal become accessable, connection to a port may not be possible. Again, the user must decide whether to wait or leave. Finally, if the computer, terminal, and port are available the user may log on. After log on the user (terminal) and computer operate independently of the other terminals and computers. The user

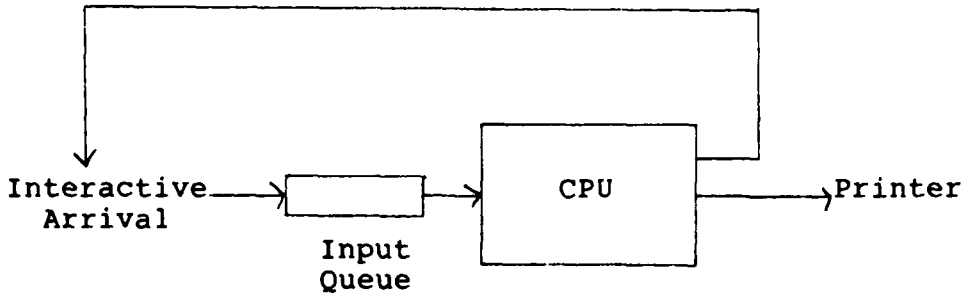


Figure 3.2 Interactive Load.

submits 'jobs' to the one computer until the session time is complete. Printer jobs may be queued to a printer accessable to the computer.

Computers. Since the purpose of this model was not to perform any computer performance measurement or evaluation upon specific computers, the internal hardware configuration of each computer was not simulated. Hence, Chandy's analytical queueing models illustrated in Chapter One (Figures 1.2-1.4) can be modified such that a computer's central processing unit (CPU) essentially becomes a 'black box'. Figure 3.2 illustrates an interactive arrival on a computer. The user submits jobs to the CPU which after completion may return output to the terminal or printer. On most computers the user has an option when jobs are submitted whether to 'wait' for the job completion (normal interactive mode) or submit as a 'batch' job which will not output back to the terminal until the user requests it. All of AFIT's computers have this capability.

As illustrated in Figure 3.3 the batch arrivals are more straight forward than the inactive arrivals. The batch

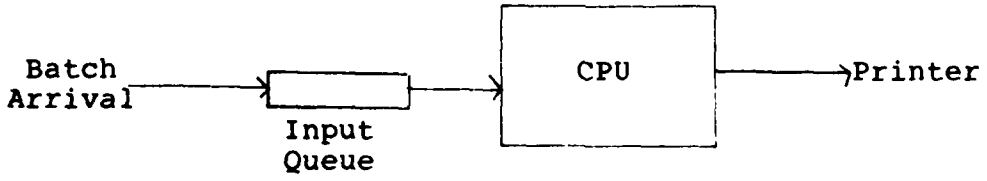


Figure 3.3 Batch Load.

job arrives and is submitted to the input queue. When there is room inside the box the job enters and after a transformation is output to a printer.

Identification of Variables

During the simulation the CPU's 'black box' transformation can be defined by keeping track of the CPU's state and the resource requirements placed by the jobs flowing through it. Table 3.2 lists these variables. The first three variables (ITRMOL, ITRMWP, ITRMAC) are utilized to track the states of the terminals currently attached to the computer. The number of terminals online, ITRMOL, is the total number of terminals currently utilizing a port into the computer. The number word processing, ITRMWP, is the total number of terminals utilizing word processing programs. These terminals do not include use of the editors for program development. The number active, ITRMAC, is equal to the total online minus the number of terminals currently waiting for a job completion. Terminals waiting for job completion (job submitted in batch mode) cannot place other interactive demands until the job completes.

Mnemonic	Description
ITRMOL	# Terminals Online
ITRMWP	# Terminals Word Processing
ITRMAC	# Terminals Active
IEXJOB	# Jobs Executing
IMEM	Total Memory Requirements of jobs Executing
ICPU	CPU Seconds Used
INOUT	I/O Seconds Used
IMPL	Multi-Program Level
TNOW	Current Time

Table 3.2 Computer State Variables.

IEXJOB is the number of jobs currently 'executing'. This variable is incremented as a job is allowed to enter the box and decremented when it exits. Total CPU memory requirements of the executing jobs are stored in IMEM. This variable is also incremented and decremented as jobs enter and exit the computer. ICPU and INOUT variables are used to store the total CPU and I/O seconds used during the statistics collection interval. Lastly, IMPL is an input variable which sets the computers multiprogram level. This multiprogram level limits the number of jobs that may enter the CPU's black box.

All these variables may be used to predict the run time as a job enters the CPU. Run time is defined as the time from when a job enters the box till the job enters the

output queue. For computers on the network which do not have the entire workload generated within AFIT (e.g. CYBER and CREATE), a system variable which may be used is the time of day. Computer performance measurement techniques (Ref 23) and computer performance modeling techniques need to be implemented on all of the network computer resources. This 'black box' may then be defined using the above variables and applying multivariate tools and analytical models on data collected.

Input Variables. The input variables of the simulation model define: the mean and standard deviation of various distributions required to drive the simulation; probability mass functions when a routing decision needs to be made; and various tables to define the complex relationships between computers, locations, and so forth. The file contains space for a mean and standard deviation even if the distribution is found to require only a mean such as exponential. This allows the distributions to be easily modified without changing the format of the variable file.

A FACTOR array was created to hold the means and standard deviations of the interactive user behavior distributions and CPU job characteristics. Interactive user behavior includes think time and number of jobs entered per session. Think time is defined as the time between one job submittal till the next job submittal. Iyengar and Chih-Chuh found this was best approximated by an empirical exponential distribution (Ref 14:310). Factors which

characterize the job behavior include CPU seconds, I/O seconds, and memory size. Factors are provided for each work class and size. An additional factor included for user behavior is BATINT which defines the probability of the user submitting the job and waiting for completion (wait mode) or submitting it as batch (batch mode). These probabilities are defined for each computer and size of work. In some computer systems, such as the CYBER, if the size becomes too large, the user is required to submit jobs in the batch mode. For small jobs the probability would be expected to be low since one of the advantages of interactive use is immediate feedback. These probabilities would also expect to change as the computers response times change. If the response time is high, then more users would be expected to use batch mode. This allows other work to be accomplished as the job runs.

Values to control the users arrivals include: the arrival rate for each input location, ARRATE; the cumulative frequency distribution for each input location, FRTABL; and the probabilities of an arrival being dial-up, DIALUP. ARRATE is the arrival rates created by the workload model for each input locations. A rate for each hour over the quarter is provided. FRTABL is the cumulative frequency distributions created by the workload model for each input location. A distribution is provided for each week of the quarter. It is used to determine computer selected, class work, and size work. DIALUP contains the probability mass function that an interactive arrival originated at an

external (dial-up) input terminal. Different functions are provided for each computer. Some computers are expected to have more dial-up arrivals than others. This probability would include the user's preception of the service provided. Dial-up arrivals would be expected to rise after normal work/school hours.

Input variables define the printers' speed, connection to the computers, and probabilities of requiring a printer resource. The printers' speeds (lines/second) are defined in an array, PRTSPD. Connections are defined by a cumulative probability mass function PRTTAB. For each computer and input location, PRTTAB contains a cumulative probability mass function which defines the printers accessable from the computer. The VAX 11/780 has only a printer/plotter connected. Therefore, the VAX 11/780 printer/plotter would be selected for all input locations. The CYBER has three printer/plotters connected in building 640, one printer connected in building 125, and one printer connected in building 641. The selection, therefore, is more complex and is dependent upon the input location.

Other variables define the probabilities of the jobs requiring a printer resource. ONEJOB is the probability that the last job submitted by an interactive user will require a printer. This probability is expected to be higher than at any other times since the last job submitted by an interactive user many times is a print of the program for later debug or a program completion which requires print

for later turn in at a class. WPPROB contains the probabilities for each class that a printer resource is required. As the complexity of a program increases within a class, more print requests are expected. Within the classes the probabilities would be expected to change. For example, a small FORTRAN program would not require much print time. However, SLAM programs require many runs with large print times.

If the print request originated at a dial-up device, printer output is expected to sometimes return to this device. This probability is defined by the variable DUPRT. In this case the speed is defined in terms of the baud rate of the connection between the network and the dial-up device. The printer time is calculated using a mean (PCPRTM) and a standard deviation (PCPRTS).

The schedules of the resources also need to be defined for the computers and input locations. SCHBAT contains the weekly schedule for the batch locations while SCHCPU contains the weekly schedule for the computers. Additionally, the card readers and printers need to be identified to open/close as the computers and locations open/close. IPRGAT identifies the printers to close when a computer is down or location is closed. ICRGAT identifies the card readers to open or close depending on the status of the computers or the input locations.

Various time delays need to be introduced as events flow through the system. These are provided in the form of a mean and standard deviation. Later analysis will need to

be performed to find the appropriate distributions. For this model's development, a normal distribution is assumed. These variables' names follow each individual condition. If a computer is down, the arrival is rescheduled using the variables CPULTM and CPULTS. If a terminal is not available, the arrival is rescheduled using the variables TRMLTM and TRMLTS. A batch operator inputs a job using the variables OPRMN and OPRSTD. A user logs on the system using variables XLOGMN and XLOGST. If a user decides not to wait for a port, the user's return is rescheduled using the variables WAITMN and WAITST.

The points where an interactive user would decide to wait or leave are defined by a probability mass function. TRMWAT contains the probabilities that a user will wait if a terminal is not available. It is defined for each input location and number currently waiting. For example, a user would probably not wait if 5 users were already waiting at an input location with only 5 terminals. However, if the location contains 100 terminals and 5 users were waiting, the user would wait with a higher probability.

Network Model Translation

After the conceptualization of the system has been completed with the complex interrelationships the system can be translated into a simulation language. This translation is greatly influenced and constrained by the language chosen. The 'world view' begins to take the shape of the structure of the language--the categorization of the

resources, activities, and processes.

SLAM allows two different methods to create a discrete simulation which Prisker divides into 'network' structure and 'discrete' structure. The 'network' structure consists of specialized nodes and branches that are used to model resources, queues for resources, activities, and entity flow. However, the 'network' structure lacks the flexibility needed to model a complex system. The 'discrete' structure provides this added flexibility by using FORTRAN subroutines and the functions defined in the 'network' structure. Both of these methods can be combined to form the network simulation model.

Because of the greater flexibility in using discrete structure, the majority of the program is written in FORTRAN with the network structure used to define only the actual resources (computers, terminals, card readers, and printers) of the system. Events generated during the simulation may enter and exit the network structure. This allows a minimum amount of code changes required when a hardware configuration is changed. The network structure, with the input variables, define the actual hardware configuration while the discrete structure handles the complex relationships between the resources. Hence, the discrete structure will not have to be modified for each configuration change.

SLAM Network Structure

The hardware configuration of the model is defined using SLAM network structure. Figure 3.4 illustrates the

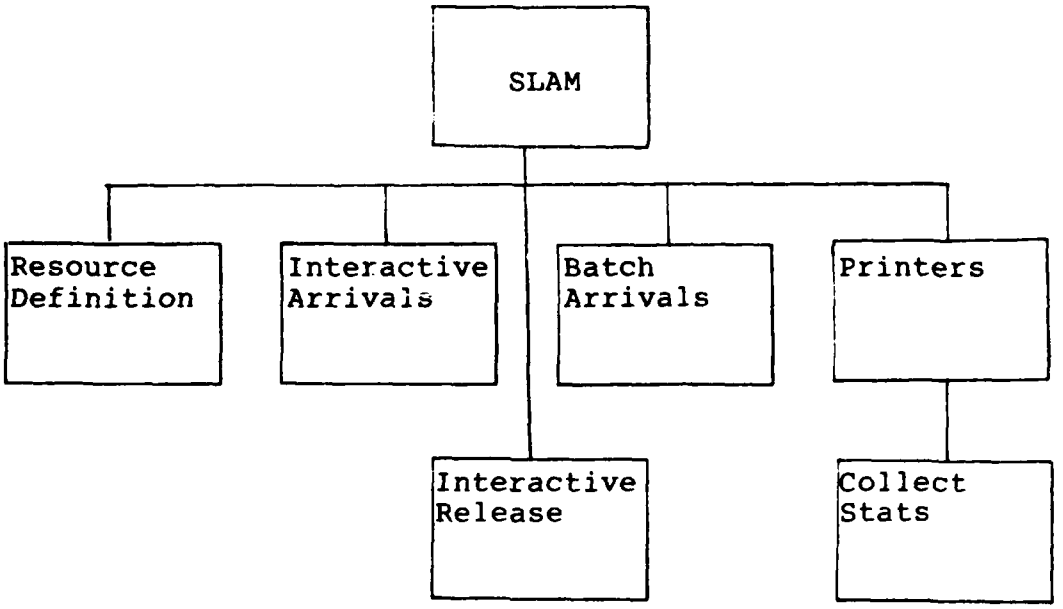


Figure 3.4 Network Structure.

modules for the network portion of the simulation -- the resource allocations, interactive arrivals and departures, batch arrivals, and printers. Batch departures are not necessary since printing is the last resource required.

At most of the modules various statistics may be collected. At batch arrivals the wait time for input may be collected. The session times may be collected after interactive release. Print time and print queue times may be collected at printers. Lastly, after a batch event finishes 'printing', the turnaround time for each computer and location may be collected.

Resource Module. The resources are defined and initialized using SLAM resource blocks, gate blocks, and global variables as shown in Table 3.3. The computers are

Resources	Defined By	Values
Computers	Global Variable	0/1
Card Readers	Gates	open/close
Printers	Resource Block	0/1
Terminals	Resource Block	0-?
Ports	Global Variable	0-?

Table 3.3 Resource Declarations.

defined by use of a single global variable which can be either equal to "1" indicating the computer is operational or equal to "0" indicating the computer is not operational. The card readers are defined by use of a gate. The SLAM gate is either opened or closed. If open, this allows the jobs (card decks) to be submitted and flow straight through the system. If the card reader is closed, jobs will still be submitted but will wait for an opening. At a gate opening all the jobs are released. The printers are defined by a resource block which can be either equal to "1" indicating the printer is operational or equal to "0" indicating the printer is not operational. The terminals are defined likewise with the number of terminals available at each terminal input location. Lastly, the ports are defined using a global variable set to the number of input ports for each computer.

Batch Arrivals. Batch arrivals enter the network through enter node 1 (Figure 3.5). If the input location/card reader is open the event immediately becomes

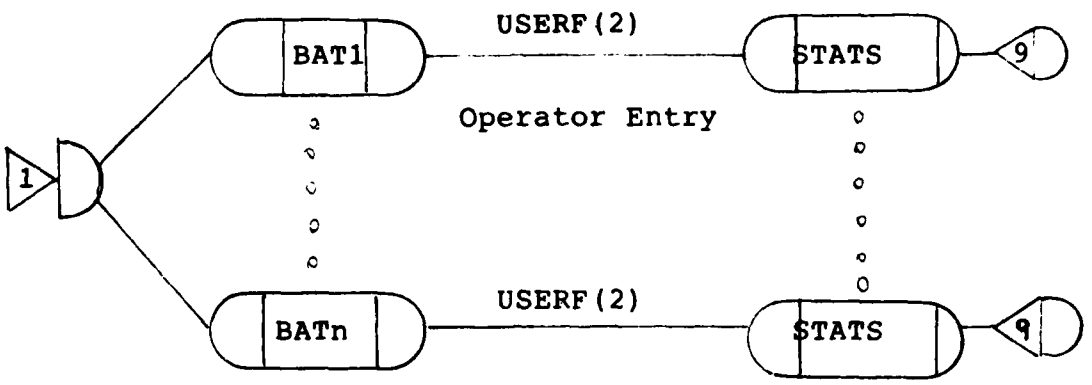


Figure 3.5 Network Batch Arrivals.

available for operator input which takes USERF(2) seconds. USERF(2) is calculated using a normal distribution with a mean of OPRMN and a standard deviation of OPRSTD. If the computer is down or the input location closed, the job may be left for subsequent input after the computer comes up or an operator opens the location. Various statistics may be collected (i.e. time waiting) before the event is routed back to discrete processing.

Interactive Arrivals. After an interactive user has decided to either seize a terminal or wait for one, discrete processing passes the event to enter node 2 as shown in Figure 3.6. The event is then passed to the AWAIT node for the input location to seize the terminal for that location. If a terminal is available, the user can then log on, USERF(1), and return to discrete processing. USERF(1) is calculated using a normal distribution with a mean of XLOGMN and a standard deviation of XLOGST. If a terminal is not

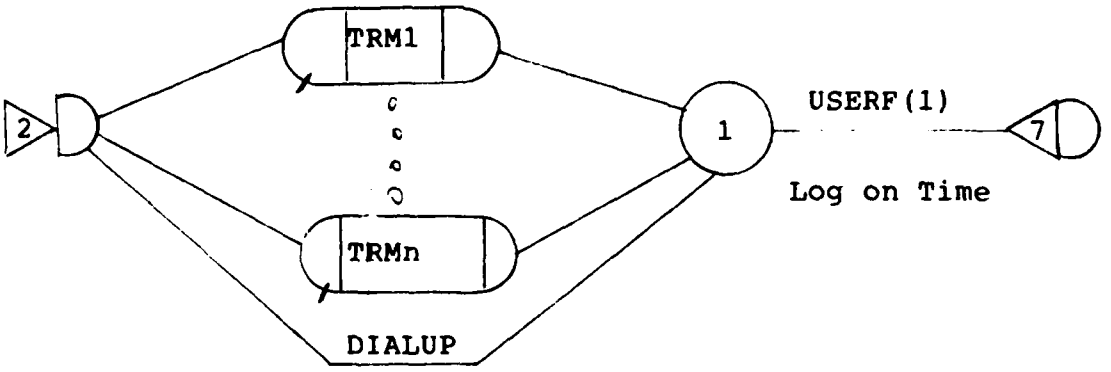


Figure 3.6 Network Interactive Arrival.

available, the user waits (at the AWAIT node) until a terminal from that location is released. If the event is a dial-up user, the event immediatly logs on using USERF(1).

Interactive Release. After the user event has completed a session or a computer is not operational, the event is routed to entry node 3 (Figure 3.7). The terminal resource is released and various statistics such as session times are collected. The event is then terminated. Additionally, if a user event cannot get a port after log on and decides not to wait, the event is routed to node 3 to release the terminal.

Printers and Collect Statistics. All requests for printers enter through node 4 as illustrated in Figure 3.8. The event is then routed to the appropriate printer queue. If the printer is available, the resource is allocated to the event and the print time, ATRIB(8), begins. ATRIB(8) contains the print time as set by discrete processing. At

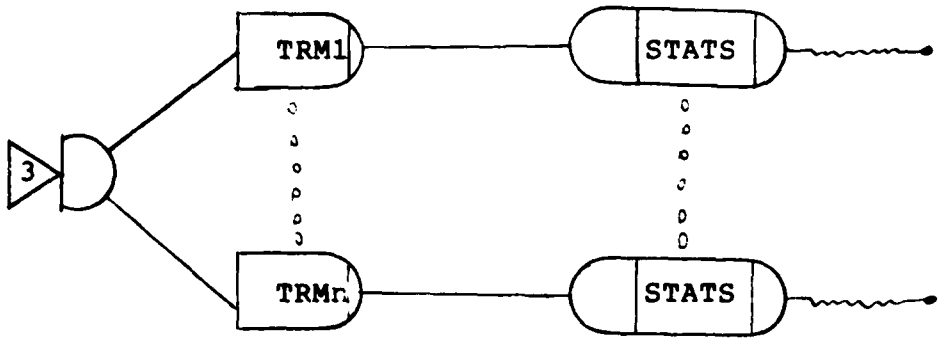


Figure 3.7 Network Interactive Release.

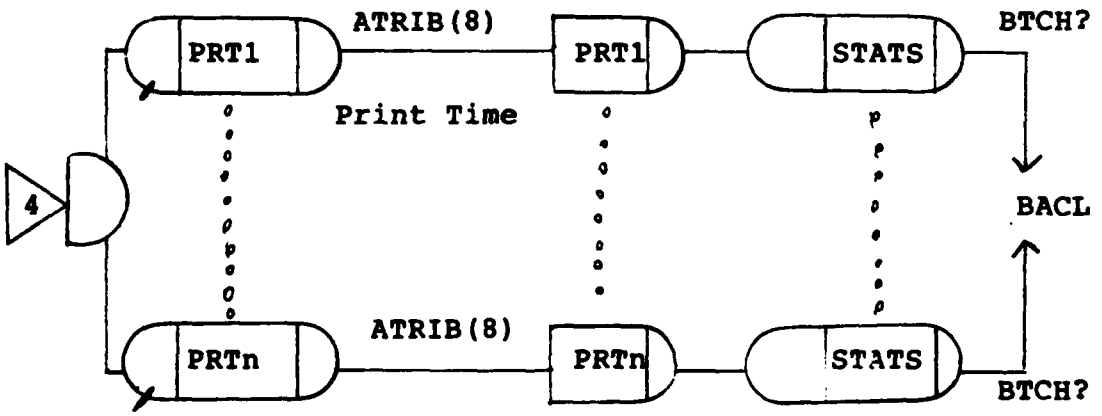


Figure 3.8 Network Printers.

the end of the print time the resource is released and another print event may start. Statistics such as print time and queue time may be collected. If the job event is from a batch location, additional statistics such as batch

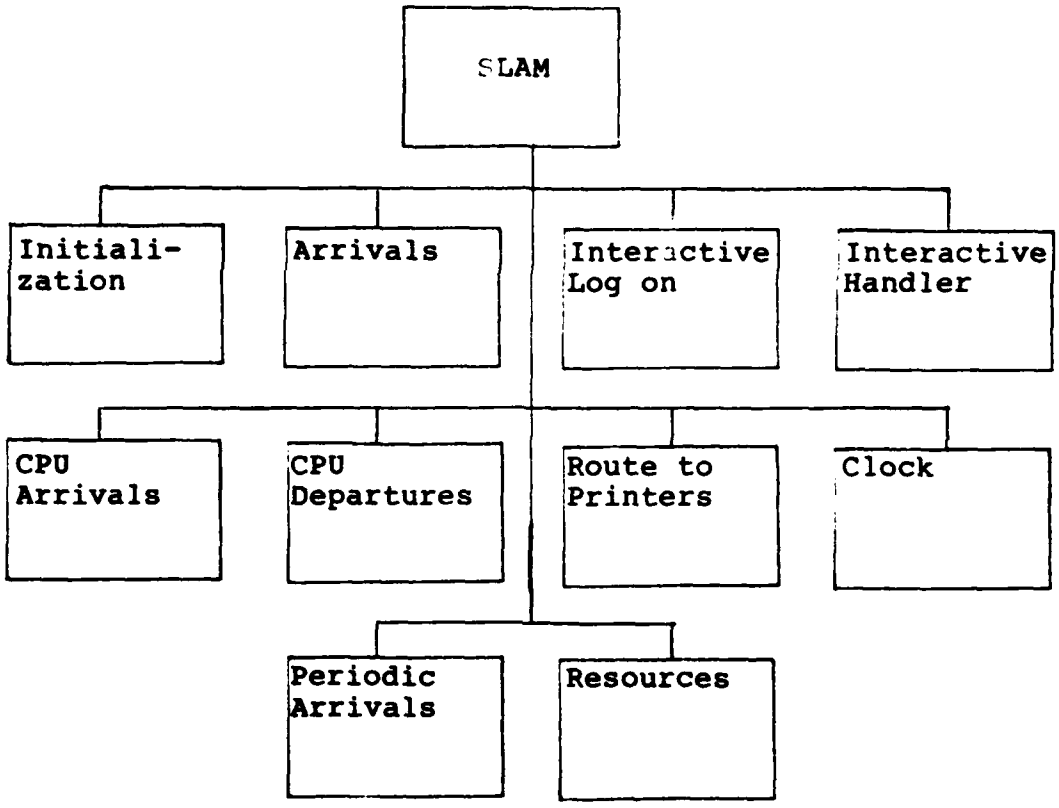


Figure 3.9 Discrete Structure.

turnaround time are collected at location BACL. Events are not returned to discrete processing.

SLAM Discrete Structure

The discrete structure of the simulation drives the simulation using the network structure and the input variables in input file FACTOR. Figure 3.9 illustrates the structure which has been divided into ten modules. One module, Arrivals, handles all regular user arrivals for both interactive and batch. Two modules, Interactive Logon and Interactive Handler, control interactive sessions. Another two modules, CPU Arrivals and CPU Departures, handle all

requests for CPU resources. Route to Printers handles all requests for printer resources. Periodic Arrivals schedules the arrivals of events from the CONSTANT input file.

Modules Clock and Resources control the day's operational schedule. Clock is scheduled every hour (3600 seconds) to update the arrival rates, cumulative frequency distributions, and so forth. Additionally, it performs end of day, week, or run processing. Daily the system resources may be scheduled to open or close during the course of the day. Resources performs the appropriate change of state.

Initialization. Figure 3.10 illustrates the start-up initialization of the network system model. All SLAM variables are initialized to 0 and system variables default values are set. Input file FACTOR is read to initialize all remaining variables. The arrival rate and cumulative frequency distribution files created by the workload model are read to initialize the arrival rate tables (ARRATE) and cumulative frequency distributions (FRTABL) for each input location.

The first events are then scheduled. A clock update is scheduled to occur in an hour. The first input arrivals are scheduled to occur for all locations using ARRATE. The resource availability tables, SCHCPU and SCHBAT, are checked to schedule any resource changes for the day. Finally, the input file, CONSTANT, is read to schedule the first periodic arrival event.

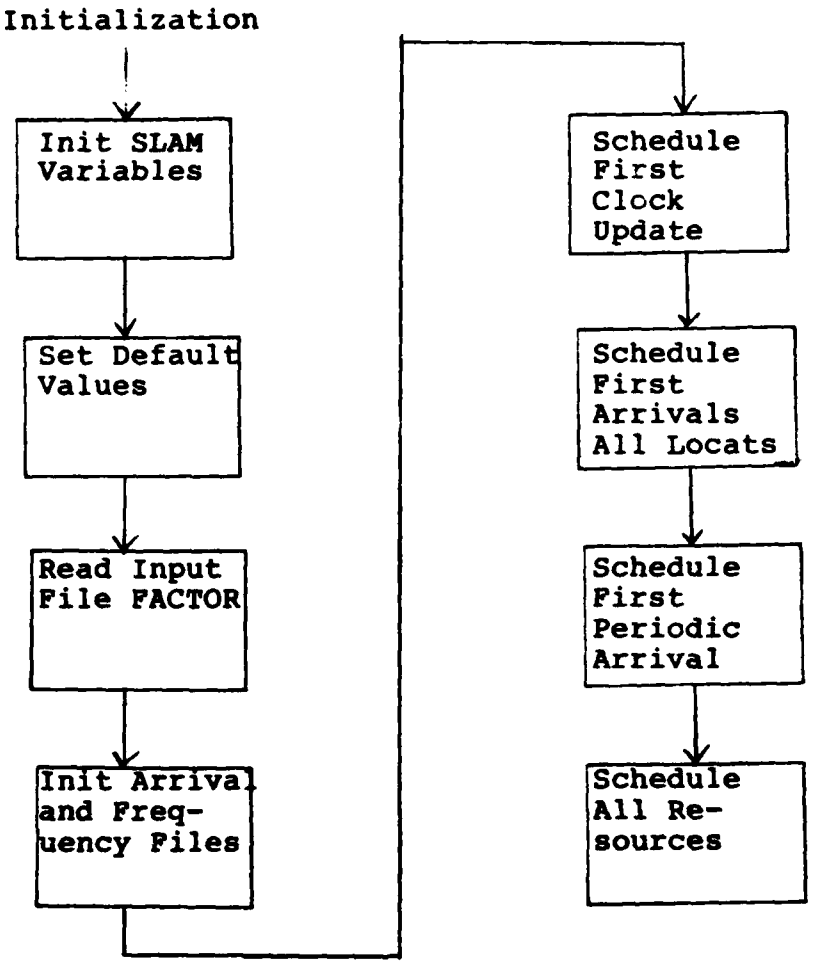


Figure 3.10 Network System Initialization.

Arrivals. All batch and interactive arrivals for all locations are processed as shown in Figure 3.11. As an event arrives from a location, the next arrival is scheduled for that location using ARRATE. The attributes of the event are then set. The FRTABL for the location is used to determine computer desired, class work, and job size. If the computer code from FRTABL is equal to the 'any' code then a computer is selected by using a probability mass function

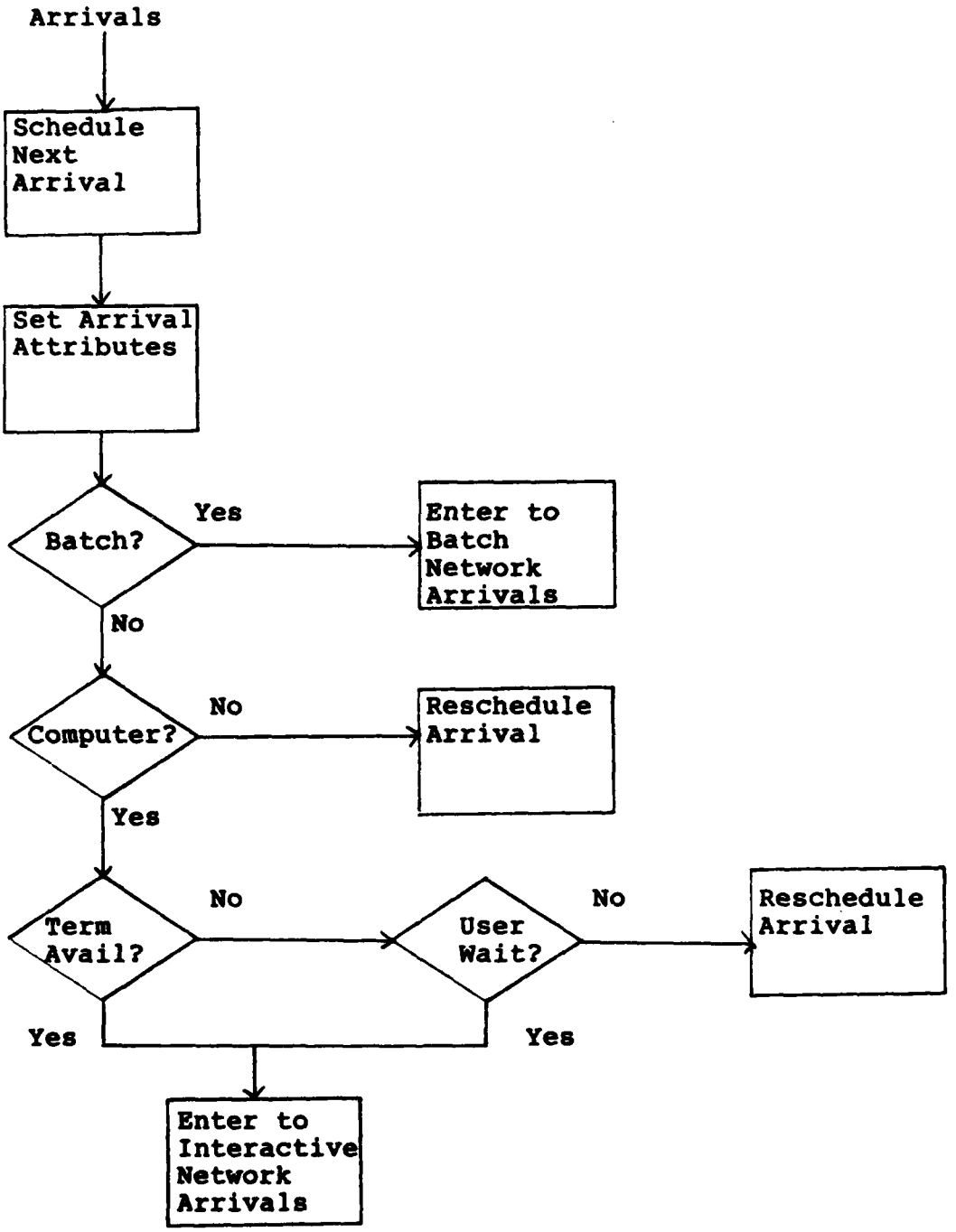


Figure 3.11 Network Arrivals.

(ANYDST) for the class work. The event may be selected for a dial-up arrival if a random number generated is less than a dial-up probability (DIALUP) for the computer and hour of day.

If it is a batch arrival, the event is then entered into the network structure. Otherwise, a check is made to see if the computer is operational. If not, the event is rescheduled to arrive. A check is then made to see if a terminal is available at the input location. If so, then the event can be routed into the network structure. Otherwise, a random number is generated and, if less than the probability that the user will wait (WAITTR), the event is routed to the network structure. WAITTR is dependent upon the current queue length and input location. If greater than, the event is rescheduled.

Interactive Log On. Figure 3.12 illustrates the Interactive Log On module. First, a check is made to see if a port is available to the computer desired. If not, a random number is generated. If the random number is less then the probability that the user will wait (WAITPR), the event is put into a wait queue. If the user will not wait, the event is entered into the network structure to release the terminal. If the port is available, the CPU states are updated (# active, # online, and # word processing terminals); the expected number of jobs the interactive user will run is set using a normal distribution with a mean and

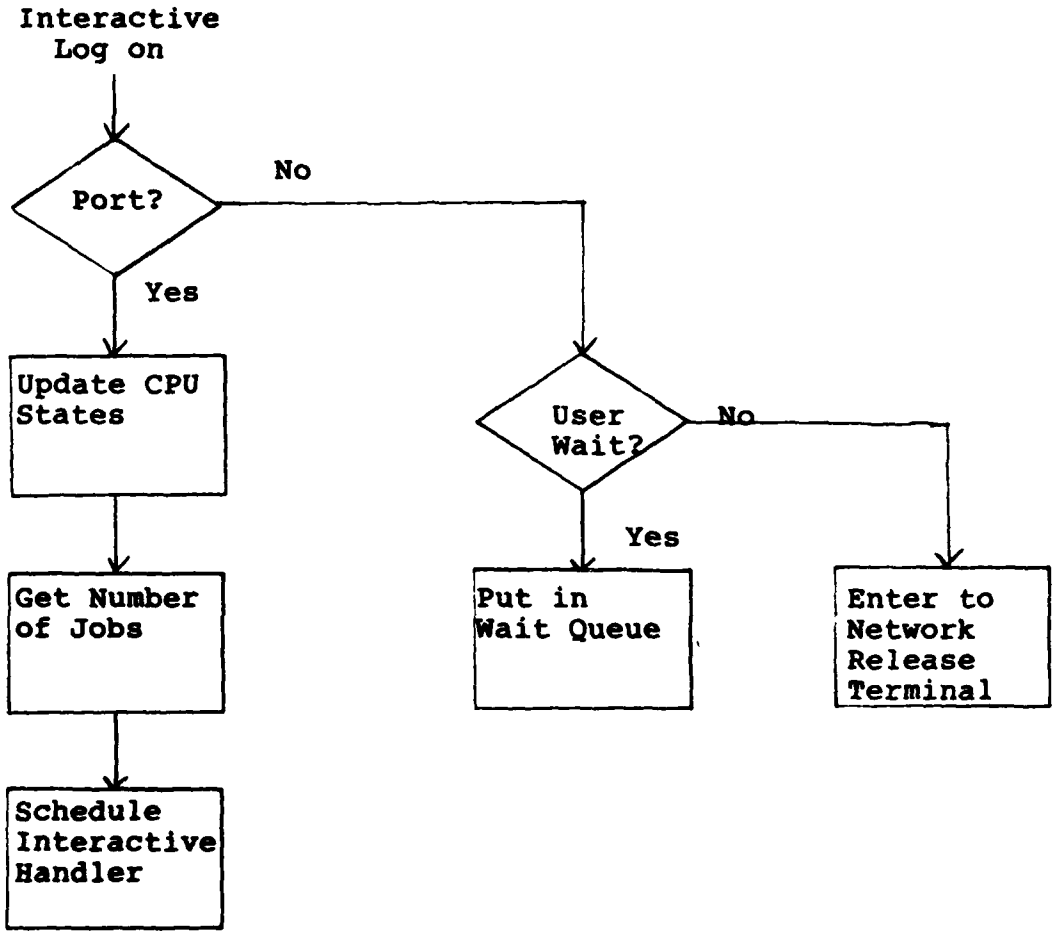


Figure 3.12 Interactive Log On.

standard deviation for the class and size of work. The first think time is set and the Interactive Handler is scheduled.

Interactive Handler. The Interactive Handler is scheduled at the end of the user think time. It is also scheduled after completion of a CPU job which an interactive user submitted in 'wait' mode. Figure 3.13 illustrates the Interactive Handler. If the job had been submitted in wait

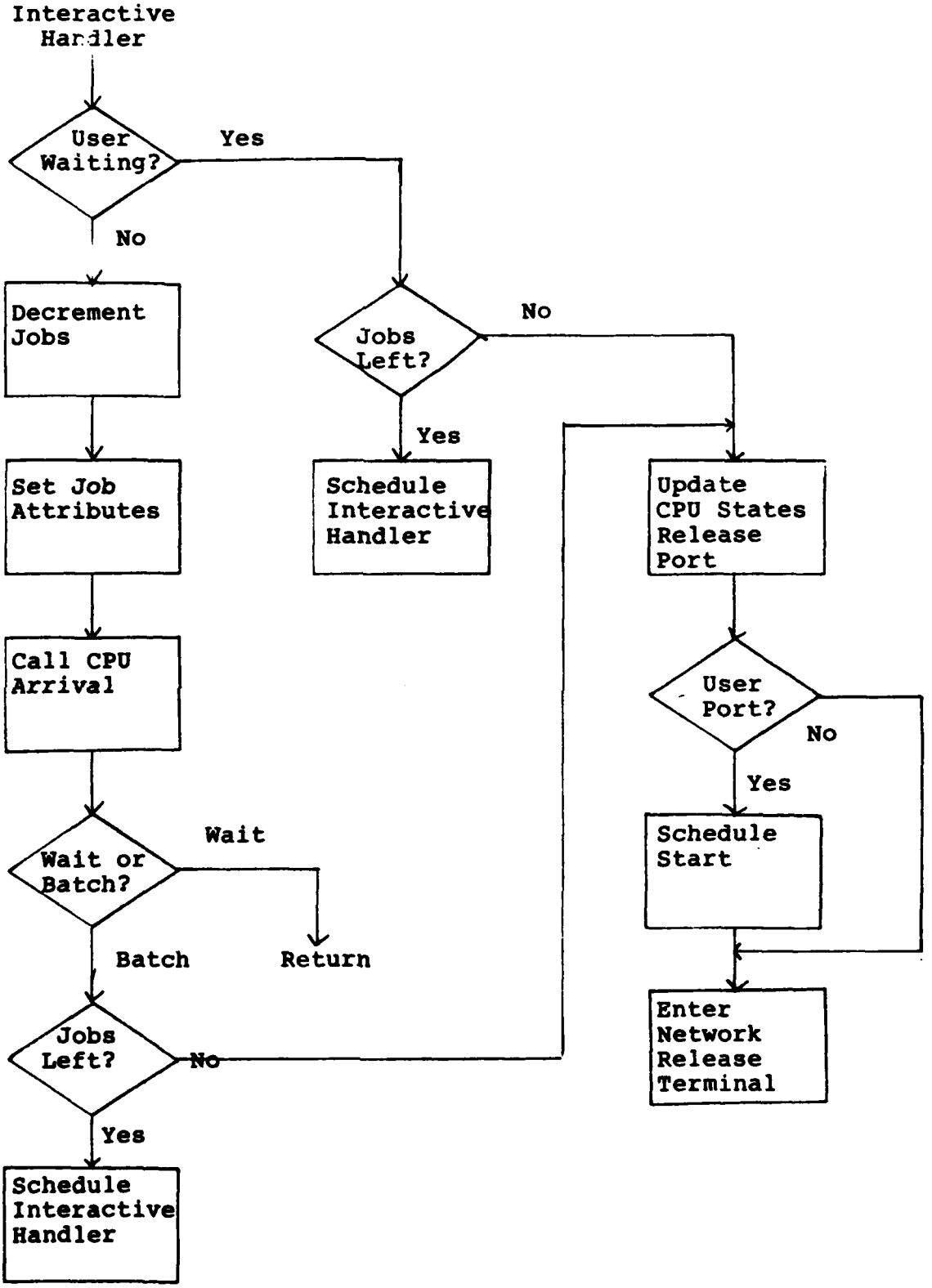


Figure 3.13 Interactive Handler.

mode, a check is made to see if any jobs still need to be generated. If so, another think time is calculated and the Interactive Handler is scheduled again. Otherwise, the session is over. If the Interactive Handler had been scheduled because the end of a user think time had been reached, the job count is decremented, the job attributes are set, and the CPU Arrival module is called to submit the job.

If the job was submitted in wait mode, the Interactive Handler is done and the user will wait till the job completes. Otherwise, the job was submitted in batch mode. If more jobs are left, the think time is calculated and the Interactive Handler is scheduled again. If no jobs are left, the session is complete.

When a session is complete the CPU states are updated. A check is made to determine if any other users are waiting for a port. If so, a user is removed from the queue and started. Finally, the event is routed to the network structure to release the terminal resource.

CPU Arrivals. The Interactive Handler and the Network Batch Arrivals submit the jobs requiring CPU resources to the CPU Arrival module (Figure 3.14). The job's CPU time, memory required, and I/O time are calculated based upon the job's class and size. If this is an interactive job in wait mode, the number of active terminals is decremented. If there is room for the job in the CPU's 'execute queue', then the run time is calculated using the CPU's states as

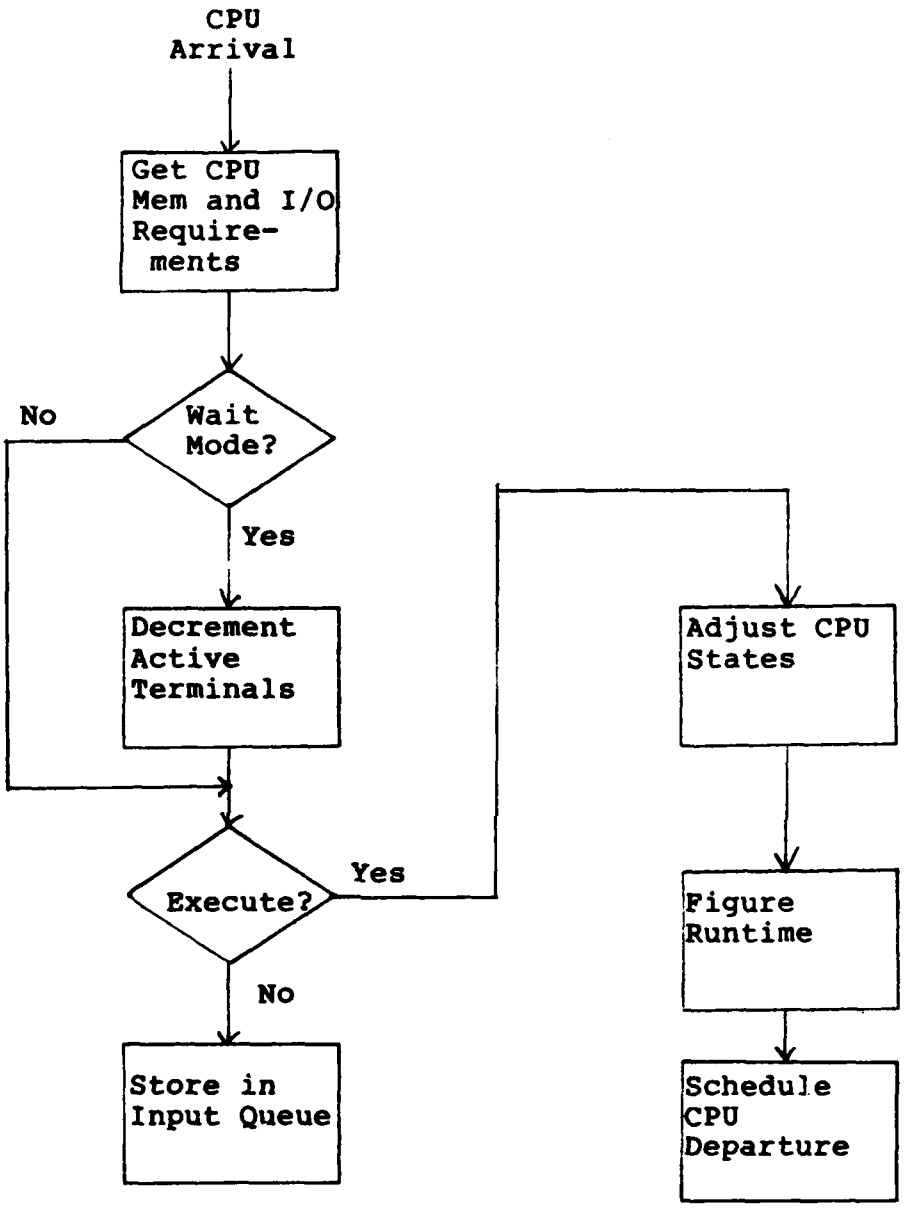


Figure 3.14 CPU Arrivals.

described earlier. The job is then scheduled for CPU departure. Otherwise, the job is stored into the CPU's 'input queue'.

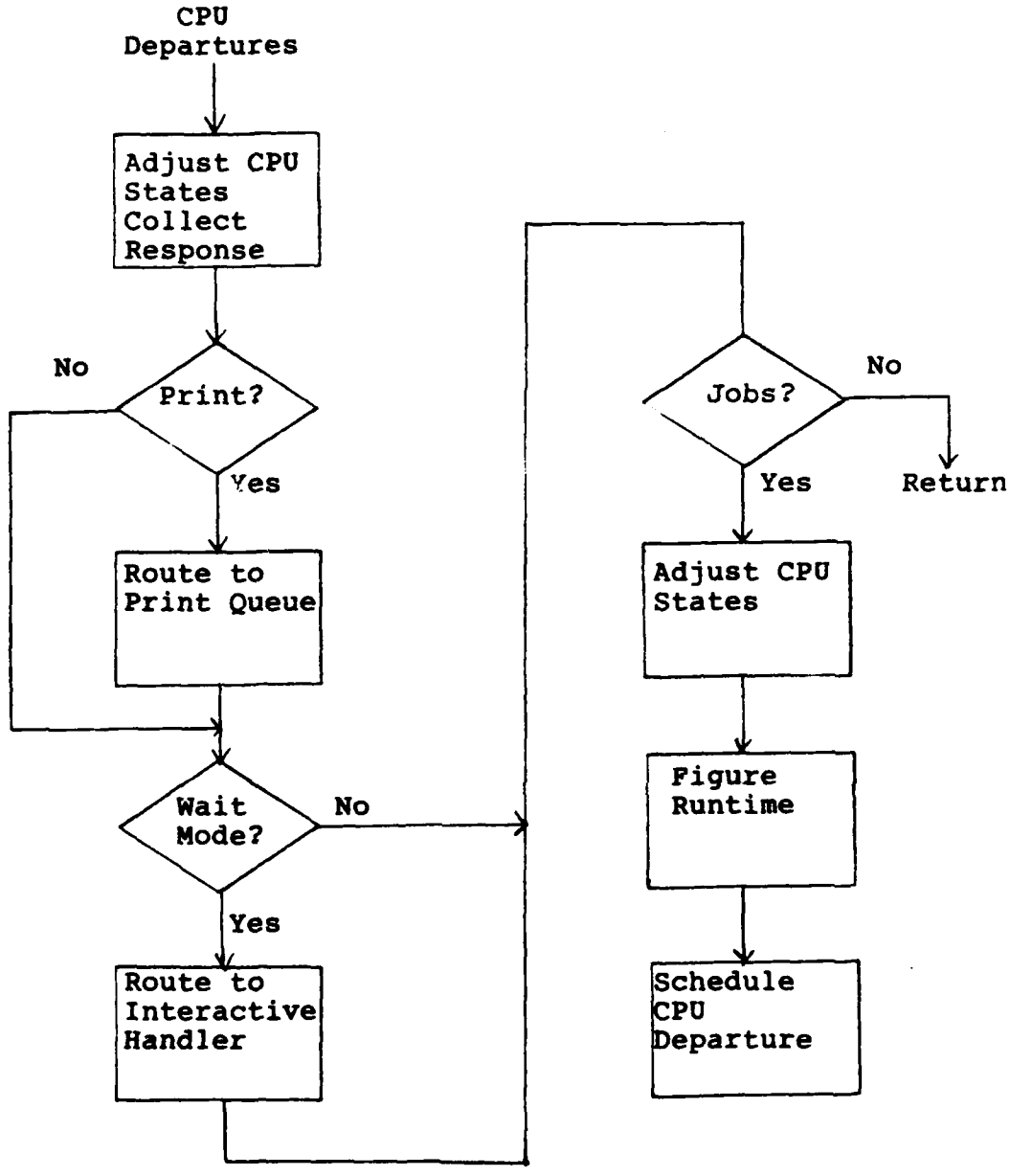


Figure 3.15 CPU Departures.

CPU Departures. Figure 3.15 illustrates the actions required when a job completes 'execution'. First, the CPU states (IEXJOB and IMEM) are decremented. Response times are collected for interactive jobs. If the job is batch or

an interactive word processing, then Route to Printer module is called to print. If the job had been submitted in interactive wait mode, the Interactive Handler module is scheduled. In any case, a check is made to see if any jobs are in the CPU's input queue. If so, then the CPU states are updated, the run time is calculated, and a job is scheduled for CPU Departure.

Route to Printers. When a job requires a printer, the Route to Printer module is called as illustrated in Figure 3.16. First, the printer needs to be determined. If this event is an interactive user on a dial-up terminal, a random number is generated. If that number is less than DUPRT a print time is calculated using a normal distribution with parameters PCPRTM and PCPRTS. The job is then immediatly entered into the network structure for printers.

If the print request is not for a dial-up printer and was submitted in user wait mode, the Interactive Handler is scheduled. Using a random number, the PRTTAB cumulative probability mass function for the computer and input location is looped through to determine an appropriate printer. The print time is calculated using APRTL and the speed of the printer, PRTSPD. Finally, the event is entered into the network printers with attribute 8 equal to the print time.

Clock. Figure 3.17 illustrates the Clock module which controls all outputs; updates the arrival rates and cumulative frequency distributions; and stops the simulation. The update time, UPTTIM, is immediately incremented by 3600. and Clock is scheduled again. If the statistics interval,

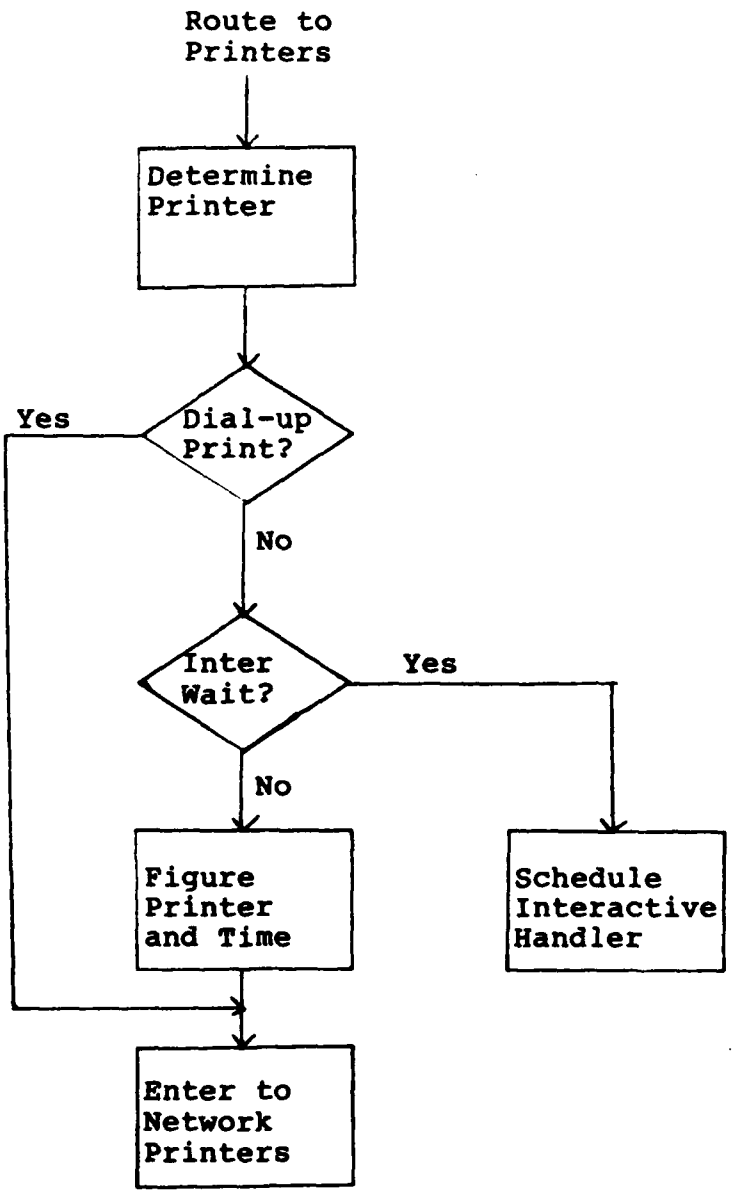


Figure 3.16 Route to Printers.

INTER, has been reached, the summary report, CPU utilizations, and I/O utilizations are output. The statistical arrays are then cleared. At the end of a day (24 hours) the arrival rate tables for all locations are updated and the

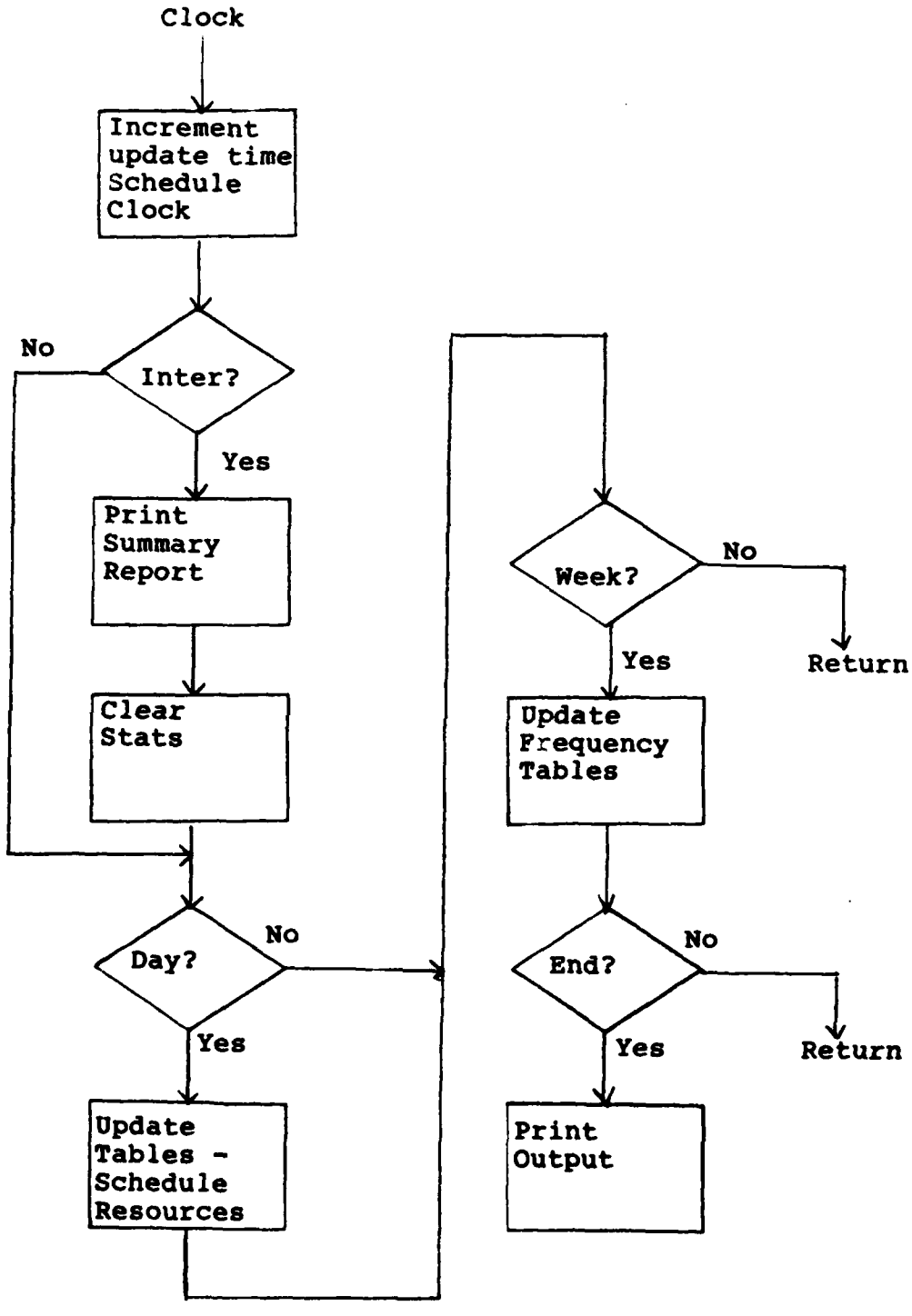


Figure 3.17 The Clock.

resource schedules (SCHCPU and SCHBAT) are processed to schedule the appropriate computers up/down and batch locations open/close.

At the end of the week (7 days) the cumulative frequency distributions for all input locations are updated. If the end of the run has been reached, then the final output is printed and the simulation ends.

Resources. The Resource module is scheduled daily to open/close resources. It is illustrated in Figure 3.18. If the request is to schedule a computer operational or not operational then the appropriate global variable is set to 1 or 0. If the computer is becoming operational, then all the card readers attached, ICRGAT, are opened. Otherwise, the attached card readers are closed.

If the request is for a batch location opening or closing, the card reader resources are opened or closed. The printers within the location as defined by IPRGAT are also opened or closed.

Verification and Validation

Verification of the model required prepration of the test input files and development of a 'log' function within the discrete structure. The function allows the output of the current time, a short description, and the attribute values of the event. The state of the computer may also be output. This function is very similar to SLAM's network structure trace option.

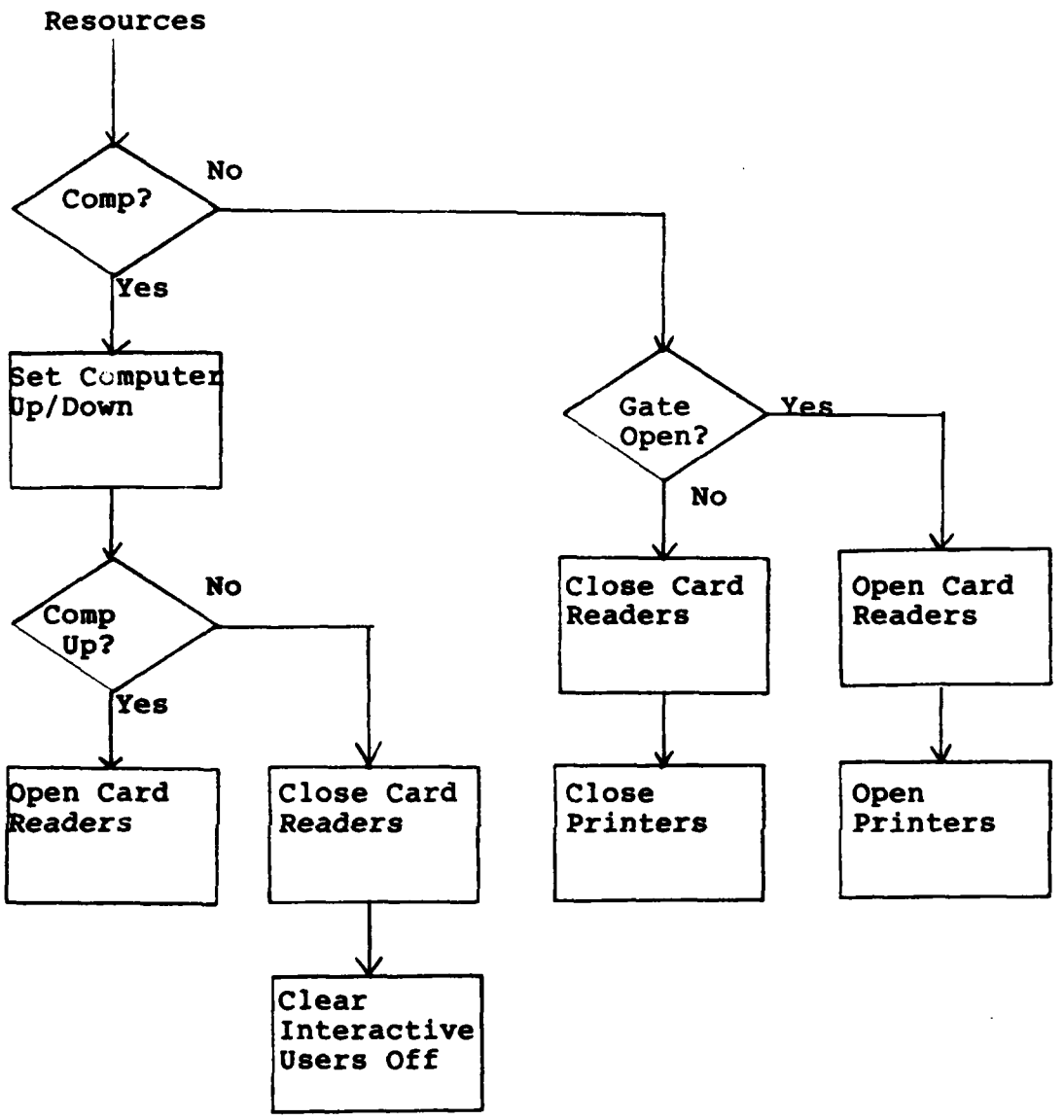


Figure 3.18 Resources Module.

For the initial verification the input files, which would be created by the workload model, were generated by another small program. This program set all of the arrival rates for all locations to one. The cumulative frequency distributions were set equal across all computers, classes,

and sizes. The input variables were set to represent reality although few of these variables can be validated. Distributions found by previous research were used when possible. Otherwise, a normal distribution was assumed.

Using the above input files it was possible to verify the routing and distribution of the users and jobs as they flowed through the simulation. The SLAM trace and the log function were utilized to debug and verify the flow of events. Resource capacities and variable values were changed occasionally to force conditions or events. All modules and conditions except the operational scheduling were verified. Finally, a test was performed to force all arrival rates to go zero. After another day's simulation no events remained in the simulation and all resource capacities were at their maximum value.

The input files created by the workload model were then used for verification. These input files provided more realistic user arrivals and requirements. Additionally, the operational schedules of the resources were tested. Again some resource capacities and variables values were changed to force conditions or events. All modules and conditions were verified.

An initial validation of the model was performed. As mentioned previously, most of the data required to validate the model does not exist. However, the real system was reevaluated to validate the basic structure and routing decisions. The output of the logging function allowed each

user to be traced throughout its entire duration in the model. This exercise was performed for every input location and computer. Each of the responses listed in Table 3.1 occurred during the model's testing. These responses have been observed on the real system.

Even without a complete validation the model's usefulness was demonstrated. For example, the operational schedules' impact on the queue lengths and turnaround times for batch jobs was obvious. During testing without operational schedules the turnaround times where small. However, after introduction of the operational schedules the turnaround times increased dramatically. If there was an AFIT standard or objective concerning turnaround time, the operational schedules for batch locations could be evaluated since they have an obvious impact on turnaround time.

After validation and implementation the model will be most useful for determining the impact of increased requirements and hardware upgrades. For example, suppose it was suggested to add more terminals to relieve the congestion and wait time at the input locations. Before adding these terminals the model could be run to determine the impact to other system parameters. The increased terminals may cause response times to degrade below acceptable standards or if the ports into a computer is not increased the user may now wait for a port instead of a terminal.

Summary

Chapter Three has presented the conceptualization of the AFIT network model followed by identification of the input variables and translation into the SLAM simulation language. The structure of each module was presented. Chapter Four presents the summary, recommendations for future research, and conclusion.

Chapter 4

SUMMARY, RECOMMENDATIONS, AND CONCLUSIONS

The primary objective of this research was to provide an initial validated dynamic system model of the AFIT ADP network. After the model is implemented, it will represent the AFIT network with its users and should be used for understanding and evaluation of the AFIT workload and hardware. Before this implementation can occur further research is necessary. Recommendations for this implementation research is presented in the recommendations for future research section in this chapter. Before presenting those recommendations, a brief summary of the model is presented.

Model Summary

A hybrid simulation methodology was used to develop the model of the AFIT network. This section summarizes the conceptualization and structure of the workload model and network model which when combined form the system model.

Workload Model. The workload model, as modeled in this research, consisted of a categorization of the users and a characterization of the workload necessary to drive the network model. The categorization of the users consisted of defining the different requirements of the users, the input locations available to the users, and other behavioral and outside influences. Workload characterization consisted of defining a resource vector or parameters required to drive the network model.

Four categories of users were defined -- student, faculty, administration, and software support. Within these categories, more specific sub-categories were defined. These categories led to the development of the input files required to define the AFIT workload. Behavioral and environmental factors included the growing influx of personal computers, interactive or batch selection, and selection of computers.

The characterization of the workload consisted of defining a resource vector X. Essentially, this vector defined the boundaries or level of detail that the workload model provided. Since the objective of this research was to provide a model which may easily be utilized to evaluate different hardware configurations, the vector could not provide specific job parameters such as actual CPU time, disk I/Os and so forth. Therefore, the vector was defined to specify computer desired, job class, and job size. The computer required also included the option of choosing any computer resource to satisfy the demand. The job classes and job sizes define the CPU time, I/O time, and memory requirements. These job classes and sizes will need to be defined by cluster analysis as described in Chapter Two and under recommendations for future research in this chapter.

The input to the model was defined using files to describe the demands placed by the categories of users. Each file defined the network input location, number of users, network resource demands, and the external factors of personal computers and interactive or batch selection. The

inputs are then transformed into arrival rates and cumulative frequency distributions to drive the network model. Lastly, periodic requirements were transformed into a separate output file.

Network Model. The AFIT network was conceptualized as consisting of three types of resources -- input locations, computers, and printers. These resources were translated into a SLAM 'network' structure. The complex interactions between these resources and the job arrivals were structured and translated into a SLAM 'discrete' structure. This division will allow all new configurations to be analyzed by modifying the network structure and the input variables to the discrete structure.

The inputs to the network model are the files created by the workload model and the input variable file. The files created by the workload model consist of arrival rates and cumulative frequency distributions for all of the network's input locations. The workload model also creates a periodic arrival file which allowed jobs to be input to the network directly. The input variable file defines the mean and standard deviation of various distributions required to drive the simulation; probability mass functions when routing decisions need to be made; and various tables to define the complex relationships between the computers, locations, and printers.

The model may execute over the time period of a school quarter or any week(s) within the quarter. A statistic

collection interval is specified at execution. This interval controls when the SLAM summary report is printed and the statistical arrays cleared. The SLAM summary report, CPU utilizations, and I/O utilizations are the only output generated.

Complete verification and an initial validation was performed. Verification of the model was achieved by the inclusion of network monitor statements and a built in 'log' function within the discrete structure. The optional 'logs' made be output during execution to follow the flow of all events. These trace mechanisms allowed all types of events to be traced through the network model. All modules were executed. Initial validation of the model was acheived by the use of the above trace mechanisms and a reevaluation of the real system. All events flow and processes were as originally conceptualized and designed.

Recommendations for Future Research

The model developed in this research provides a theoretical structure for the AFIT workload and network. Successful implementation of the model will require additional research. Data needed by the model was identified. However, this data does not exist in an appropriate form required by the model. Initial validation of the model was performed when possible but again data needed to completely validate the model was not available. Stages six through ten of Shannon's eleven stages (page 19) remain to be completed.

The last stages of the model development cannot be completed until a broader data collection plan is implemented at AFIT. The model is just one part of a complete capacity planning program. Basically, capacity planning is a performance oriented approach to data processing management in which user satisfaction is the most critical factor (Ref 5:6). An overview of capacity planning and the current situation at AFIT is presented. Lastly, enhancements to the model are suggested.

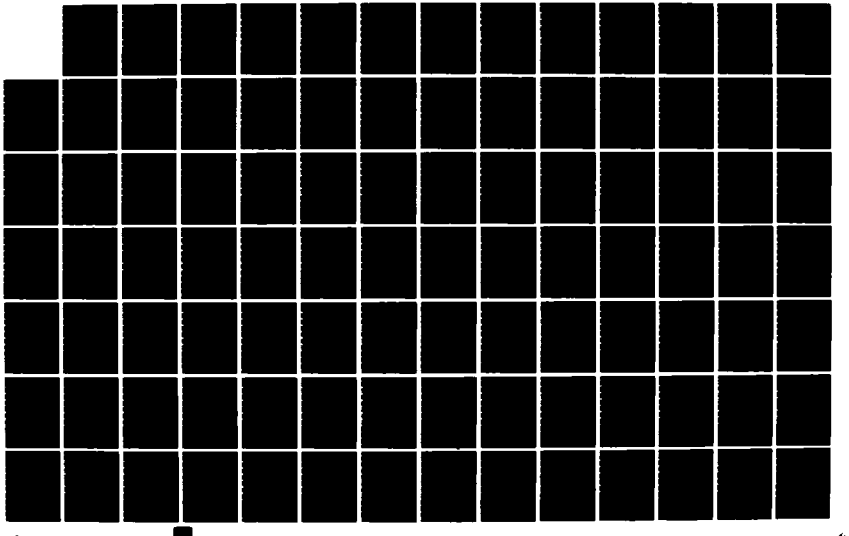
Capacity Planning. The system developed is just one component of an ADP capacity planning effort. Capacity planning is a methodology which encompasses a set of actions all geared to defining workload characterizations, forecasting workloads, current and future performance, and availability of resources (Ref 10:55). Although it was not the intent of this research to provide a review of the current capacity planning efforts at AFIT, a review is necessary for the later successful implementation of the system model.

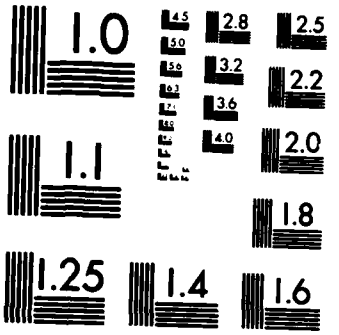
Cortada cateloged the areas that good and comprehensive capacity plans generally provide management as:

1. A description of a system's current performance. This includes utilization of individual and combined performance of all hardware and software.
2. An analysis of the current workload's characteristics, which can then be the basis for forecasting future needs.
3. A study of future workload requirements and configurations necessary to support them.
4. An analysis of response and/or turnaround time for various applications. A study of the time

AD-A141 253 DEVELOPMENT OF AN AFIT (AIR FORCE INSTITUTE OF TECHNOLOGY) ADP SYSTEM NETWORK MODEL(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. 2/3

UNCLASSIFIED S M MCDERMOTT DEC 83 AFIT/GCS/OS/83D-1 F/G 9/2 NL





it takes a particular resource to do a set of tasks.

5. A means for predicting the future performance of a configuration of hardware and software.
6. An ongoing process for providing information about the system to management, so that there are no surprises and no degradation of services.
7. A means of indicating clearly to non-DP management how well MIS managers run their departments. Ultimately, it quantifies the costs and benefits of competent DP management.
8. A way to highlight the problems caused when users and DP personnel do not participate continually and jointly in capacity planning. (Ref 10:55-56)

The best method to gain knowledge of the current system's performance and perform analysis of workload characteristics is by use of special programs or accounting packages. These programs collect data which describe the amount of resources consumed by or in support of each application program run on a computer. Three traditional users of accounting data have been ADP managers, programmers, and computer performance evaluators (Ref 16:61).

AFIT ADP managers seldom utilize any of the information gathered from the computer systems. The VAX 11/780 package collects massive amounts of data about the lengths of sessions to the number of disk I/Os. However, none of this information is processed unless a specific question is asked or a problem arises. ASD provides AFIT with a daily usage report and a monthly summary for the CYBER systems. These reports are not utilized. The CREATE and Harris 500 systems' accounting data has not been collected since April 1983. New operating systems installed were incompatible

with existing accounting packages. Of the data collected before April, the meaning of the reports is not available.

A key to being able to characterize the users' requirements on these systems is the account number. If an account number is assigned to each course, thesis, and so forth the resource requirements of each category could be determined. Current AFIT policy is to assign an account number for each AFIT student on the VAX 11/780. The student uses the same number for all work performed. CYBER account numbers are assigned as requested, but again it is unknown what type of work is being performed.

A part of capacity planning effort is systems performance management (Ref 10:56). Systems performance is responsible for making the current configurations work well. It is the main source of input into capacity planning. Capacity planning is concerned with understanding what the resources are and predicting what will be needed in the future. Performance management is responsible for three areas. First, objectives should be established for the various workloads handled by the system. For example, response time for each transaction should be 3 seconds or less, or batch turnaround should be 30 minutes. By comparing actual system performance against these objectives, a basis for future tuning is provided. AFIT ACD has not established any objectives. Second, priorities for types of jobs going through the systems should be established. Establishing priorities allows the processing of the most critical workload during peak hours. AFIT does

not establish priorities for their computers. ASD establishes batch priorities on the CYBER. Lastly, utilization of the various components of the system should be gathered. This involves capturing information on resource utilization using the accounting packages described and other software or hardware monitors.

In summary, no formal structured capacity planning effort exists at AFIT. Only after implementation of an effort will complete validation of the AFIT network system model be possible. Prediction of future performance of workload and various hardware configurations cannot be made until the current systems performance and workload are known.

Model Enhancements. The models developed in this research provide a theoretical structure of the AFIT ADP network. These models, like any software effort, are never totally complete and there are several areas in which further research would be useful. These enhancements are not necessary for a successful implementation of the models although some may be helpful for validation.

Currently, the workload model provides the inputs required to drive the network model. Aside from the printing of the input files, arrival rates, and cumulative frequency distributions, no other calculations or output are provided. The model could be enhanced to provide: 1) calculations of the total workload requirements with respect to school, computers, job classes and so forth; 2) analysis of personal computer usage data collected; 3) reports defined and pro-

duced to represent the arrival rates and distributions in a form more easily understood.

The outputs of the network model currently consist in the format provided by the SLAM summary report and CPU utilizations. While this output contains all the information required for an analysis effort, the format is not desireable. The report contains a multitude of statistics which may not be used. Again, reports could be defined to reduce the data and present it in a format more easily understood. Another possibility, besides the printed output, is creating a output file which contains all the data desired. This data could then be used for report generation or statistical analysis using other programs such as Statistical Package for Social Sciences (SPSS) (Ref 17).

Conclusion

In conclusion, the objective of this research was accomplished. A theoretical model of the AFIT network has been developed and initial verification and validation was accomplished. As part of a complete capacity planning effort, the model can provide additional information for ADP managers to use in conjunction with intuition, judgement, and experience to evaluate policies, requirements, and added resources. Additionally, the model is useful as an aid to gain a better understanding of the overall network system. Finally, recommendations have been made for further research to integrate the model into a complete capacity planning effort.

SELECTED BIBLIOGRAPHY

1. Abrams, Marshall D. and Siegfried Treu. "A Methodology for Interactive Computer Service Measurement," Communications ACM, 20 (12):93[illegible] (Dec 1977).

2. Agrawala, A.K., J.M. Mohr and R.M. Bryant. "An Approach to the Workload Characterization Problem," Computer, 9 (6):18 (Jun 1976).

3. Allen, Arnold O., "Queueing Models of Computer Systems," Computer, 13 (4):13 (Apr 1980).

4. AFIT/ACD. Data Automation Action Plan. Wright-Patterson AFB, Ohio: Air Force Institute of Technology, Mar 83.

5. Bronner, L., "Overview of the Capacity Planning Process for Production Data Processing," IBM Systems Journal, 19 (1):4 (Jan 1980).

6. Borovits, Israel and Seev Neumann. Computer Systems Performance Evaluation: Criteria, Measurement, Techniques, and Costs. Lexington,Massachusetts: D.C. Heath and Company, 1979.

7. Buzen, J.P. The Role of Computer Performance Modeling. Lincoln, MA: BGS Systems, Inc., undated.

8. Chandy, K. Mani and Charles H. Sauer. "Approximate Methods for Analyzing Queueing Network Models of Computing Systems," ACM Computing Surveys, 10 (3):281 (Sep 1978).

9. Chlamtac, Imrich and William R. Franta. "A Generalized Simulator for Computer Networks," Simulation, 39 (4):123 (Oct 1982).

10. Cortada, James W. Managing DP Hardware: Capacity Planning, Cost Justification, Availability, and Energy Management. Englewood Cliffs, N.J.: Prentice Hall, Inc., 1983.

11. Ferrari, Domenico. Computer Systems Performance Evaluation. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1978.

12. Graham, G. Scott. "Queueing Network Models of Computer System Performance -- Guest Editor's Overview," ACM Computing Surveys, 10 (3):219 (Sept 1978).

13. Hartrum, Thomas C. and Jimmy W Thompson. The Application of Clustering Techniques to Computer Performance Modeling. Wright-Patterson AFB, Ohio: Air Force Institute of Technology, School of Engineering, October 1979.

14. Iyengar, S. Sitharama and Wendy Chih-Chun. "Performance Statistics of a Time-Sharing Computer Network," Computer Networks, 6 (5):303 (Nov 1982).

15. Markowitz, Harry M. "Simulator Design and Programming" in Computer Performance Modeling Handbook, edited by Stephen S. Lavenberg. New York: Academic Press, 1983.

16. Morris, Michael F. and Paul F. Roth. Computer Performance Evaluation: Tools and Techniques for Effective Analysis. New York: Van Nostrand Reinhold Company, 1982.

17. Nie, Norman H., et al. Statistical Package for the Social Sciences. New York: McGraw-Hill Book Company, 1975.

18. Nguyen, H.C. et al. "The Role of Detailed Simulation in Capacity Planning," IBM Systems Journal, 19 (1):81 (Jan 1980).

19. O'Neill, Paul and Avis O'Neill. "Performance Statistics of a Time Sharing Network at a Small University," Communications ACM, 23 (1):10 (Jan 1980).

20. Pritsker, A. Alan and Claude D. Pegden. Introduction to Simulation and SLAM. New York: Halsted Press, 1979.

21. Shannon, Robert E. Systems Simulation: The Art and Science. Englewood Cliffs, N.J.: Prentice-Hall Inc., 1975.

22. Spagins, John. "Analytical Queueing Models - Guest Editor's Introduction," Computer, 13 (4):9 (Apr 1980).

23. Svobodova, Liba. Computer Performance Measurement and Evaluation Methods: Analysis and Applications. New York: American Elsevier Publishing Company, Inc., 1976.

APPENDIX A

WORKLOAD MODEL USER'S GUIDE

Introduction

The AFIT workload model creates output files containing arrival rates and cumulative frequency distributions for subsequent use by the network model. These output files characterize the workload for each network input location at AFIT. The arrival rates are provided for each hour over the course of a school quarter while the cumulative frequency distributions are provided for each week. A quarter consists of ten weeks of classes and one week of finals for a total of eleven weeks.

The workload is characterized by a transformation of the nine input files describing the AFIT workload into the output files described above. These input files are divided into four categories -- student, faculty/administration, software development, and periodic requirements. For the first three categories, arrival rates and cumulative frequency distributions are calculated and processed separately. For the last category, an output file of actual requirements is created. All of the groups batch requirements are combined to calculate the output files for the batch locations.

Configuration and control of the model is provided through the use of a factor file and FORTRAN namelist inputs. Most variables are assigned default values. The factor file contains all distributions used by the model while the namelist inputs control the dimension of the arrays describing the computers, work classes, work sizes, and number of input locations.

Filename	Format	Description
CRSWRKx	1	Student course requirements.
THRWRKx	1	Student thesis requirements.
STUDWPx	2	Student word processing requirements.
FACRCHx	1	Faculty research requirements.
FACTWPx	2	Faculty word processing requirements.
DAYREQx	3	Daily administration requirements.
SFTDEVx	3	Software development shop requirements.
WEKREQx	4	Weekly periodic requirements.
PERREQx	4	Quarterly periodic requirements.

x = unique identifier

Table A.1 Workload Input Files.

This user's guide used in conjunction with the Workload Model Maintenance Guide provides all the data required to operate the model. Descriptions of all the input files are provided. The input variables description and format are provided. Lastly, the CDC CYBER job control inputs, error conditions, and limitations of the model are reviewed.

Inputs

Input files to the AFIT workload model consist of nine files to describe the workload, a factor file to initialize variables, and namelist inputs. The nine input files are listed in Table A.1. The file names consist of seven characters. The first six characters remain constant. The seventh character is an unique identifier provided by the user as described later. The format number identifies each file's format as described below.

Format 1 Input Files. CRSWRK, THSWRK, and FACRCH format description is provided in Table A.2. The input location is an integer field which identifies the workload's

Format	Description	Values
I1,1X	Input Location	1-3 1-Building 125 2-Building 640 3-Building 641
A8,1X	Description	any characters
4A1,1X	Quarters Offered	blank or 'X'
4I2	Users per Quarter	integer
11(1X,3A1)	Work per Week	
	Computer	Computer ID Table
	Class	Class ID Table
	Size	Size ID Table
F5.2	PC Factor	0.0-1.0
F5.2	Batch Factor	0.0-1.0

Example: 1 EE4.51 X X 21 025 0 CA1 HE4 AB5...VC2 0.15 0.20

Table A.2 Format 1 Input File Description.

Computers		Classes		Sizes	
COMPNM	COMPID	CLASNM	CLASID	SIZNMS	SIZIDS
CYBER	C	BASIC LG	A	<20K	1
CREATE	R	DBMS	B	20K-40K	2
HAR500	H	SPSS	C	40K-100K	3
VAX	V	SLAM	D	100-200K	4
ANY	A	CANNED	E	200-300K	5
UNDEF		WORDPROC	F	OVR 300K	6

Table A.3 Computer, Class, and Size Descriptions.

expected input location. The description field is for identification only and is not used by the model. An 'X' should be input for each school quarter that the workload is applicable. The number of users per quarter must be identified for each quarter identified above. The workload expected per week is identified by alpha codes for the computers, classes, and sizes. Table A.3 identifies the current descriptions and codes. The PC factor ranges from 0.0 to 1.0 and identifies the portion of the workload re-

Format	Description	Values
I1,1X	Input Location	1-3 1-Building 125 2-Building 640 3-Building 641
A2	Quarter	WI, SP, SU, FA, or AL
I5	Users per Quarter	integer
F6.3	Percent	0.0-1.0
11F4.1	Load per Week	0.0-1.0
F5.2	PC Factor	0.0-1.0

Example: 2 WI 750 0.75 0.15 0.11... 0.25

Table A.4 Format 2 Input File Description.

quirements specified to be satisfied by the use of personal computers. Lastly, the batch factor also ranges from 0.0 to 1.0 and identifies the portion of the workload requirements specified to be satisfied by use of batch inputs instead of interactive.

Format 2 Input Files. STUDWP and FACTWP format description is provided in Table A.4. Each record identifies a word processing requirement for each quarter. The input location and PC factor are the same as format 1. The quarter is identified by a two character ID followed by the number of users expected. The percent field is used to identify the percentage of users which will probably utilize the network's word processing capabilities. The load per week values range from 0.0 to 1.0 and are the percentage of the probable users expected to require word processing capabilities for each week.

Format 3 Input Files. SFTDEV and DAYREQ format description is provided in Table A.5. These files describe

Format	Description	Values
I1,1X	Input Location	1-3 1-Building 125 2-Building 640 3-Building 641
A10,1X	Description	any characters
A2,1X	Quarter	WI,SP,SU,FA, or AL
A1	Computer	Computer ID Table
A1	Class	Class ID Table
A1	Size	Size ID Table
I4	Number of Users	integer
F6.3	Batch Factor	0.0-1.0

Example: 3 PROJECT X SP CD5 12 0.10

Table A.5 Format 3 Input File Description.

the daily constant requirements needed by the software development shop, faculty, and administration. Again the input location and description fields are the same as format 1. The quarter and number of users are identified like format 2 files. The computer, class, and size are identified by the alpha codes in Table A.3. Lastly, the batch factor must be provided.

Format 4 Input Files. WEKREQ and PERREQ format description is provided in Table A.6. These files describe the periodic requirements which will be output to file CNST. The input location for this format corresponds to the input location of the network model. The description, quarter, computer, class, and size are identical to the files in format 3. For WEKREQ the day number is the day of the week (1-Monday...7-Sunday) that input is expected. For PERREQ the day number is the day of the quarter (1-77) that input

Format	Description	Values
I2,1X	Input Location	1-10
A10,1X	Description	any characters
A2,1X	Quarter	WI,SP,SU,FA, or AL
A1	Computer	Computer ID Table
A1	Class	Class ID Table
A1	Size	Size ID Table
I5	Day Number	integer
F8.4	Time of Day	real

Example: 10 SCH RPT A AL HF6 76 22.5

Table A.6 Format 4 Input File Description.

is expected. The time of day, expressed in hours, is the time input is expected.

Factor File. A factor file (FACTOR) must be created to initialize the model's distributions and factors. This file is illustrated in Table A.7. The first record contains the NUMCPU, NUMCLS, and NUMSIZ defined by the FACTOR file. The daily distributions of the student, faculty/administration, and batch requirements are provided next. The hourly distributions for the three types of locations are provided. Next, a SESFACT, PCFACT, and RUNFACT for each class and size is provided. SESFACT is the expected number of interactive sessions required. PCFACT identifies whether a personal computer can be used. RUNFACT is the expected number of batch runs required for each requirement. There must be NUMCLSXNUMSIZ records.

BAFACT identifies whether the computer has batch capability. There must be NUMCPU records. WPFACT contains the word processing distributions over the computers by size. There must be NUMCPU-1 records -- word processing is not

Format	Description
3I5	NUMCPU,NUMCLS,NUMSIZ
	**** DAILY DISTRIBUTIONS ***
10X,7F10.4	DAYSST (Student)
10X,7F10.4	DAYSAD (Faculty/Admin)
10X,7F10.4	DAYSBA (Batch)
	*** HOURLY DISTRIBUTIONS ***
3(10X,8F8.4)	HOURST (Student)
3(10X,8F8.4)	HOURAD (Faculty/Admin)
3(10X,8F8.4)	HOURBA (Batch)
	*** FACTORS ***
10X,3F10.4	CLS SIZE SESFACT PCFACT RUNFACT
	. . . . .
	. . . . .
	**** BATCH FACTOR ***
10X,F10.4	CPU BAFACT
	. .
	. .
	**** WORD PROCESSING FACTORS ***
10X,10.4	CPU SIZ WPFACT
	. . .
	. . .

Table A.7 FACTOR File Format.

distributed to the 'any' computer. Table A.8 is an example of a FACTOR file.

Namelist Inputs. Three namelist inputs may be provided to change the model's default values. The first, FLAGNM, must be input. This record defines the printer output desired by the use of 'flags'. These flags and outputs are described in the Output section of this guide. SETDIM is set to true if a namelist RDDIM follows. SETNMS is set to true if namelist RSNMS is included.

```
   5   6   5
xxxx DAILY DISTRIBUTIONS xxxx
STUDENT      0.15      0.20      0.20      0.20      0.15      0.08      0.02
ADMIN        0.20      0.16      0.20      0.20      0.15      0.03      0.06
BATCH        0.25      0.15      0.15      0.15      0.25      0.05      0.00
xxx HOURLY DISTRIBUTIONS xxx
STUDENT    0.02    0.01    0.01    0.01    0.01    0.01    0.01    0.04
           0.04    0.04    0.07    0.07    0.11    0.07    0.07    0.11
           0.08    0.05    0.05    0.03    0.03    0.02    0.02    0.02
ADMIN      0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.01
           0.05    0.15    0.10    0.10    0.10    0.10    0.10    0.10
           0.10    0.06    0.02    0.01    0.00    0.00    0.00    0.00
BATCH      0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
           0.02    0.03    0.10    0.15    0.10    0.10    0.10    0.05
           0.20    0.10    0.03    0.02    0.00    0.00    0.00    0.00
xxx FACTORS xxx
   A  1       3.0       0.10      5.0
   A  2       3.0       0.10      5.0
   A  3       3.0       0.10      5.0
   A  4       3.0       0.10      5.0
   A  5       3.0       0.10      5.0
   A  6       3.0       0.10      5.0
   B  1       3.0       0.10      5.0
   B  2       3.0       0.10      5.0
   B  3       3.0       0.10      5.0
   B  4       3.0       0.10      5.0
   B  5       3.0       0.10      5.0
   B  6       3.0       0.10      5.0
   C  1       3.0       0.10      5.0
   C  2       3.0       0.10      5.0
   C  3       3.0       0.10      5.0
   C  4       3.0       0.10      5.0
   C  5       3.0       0.10      5.0
   C  6       3.0       0.10      5.0
   .  .        .         .         .
   .  .        .         .         .
   .  .        .         .         .
   F  4       3.0       0.10      5.0
   F  5       3.0       0.10      5.0
   F  6       3.0       0.10      5.0
xxx BATCH FACTOR xxx
CYBER          1.0
CREATE         1.0
HAR 500        1.0
SSC            0.0
ANY            1.0
xxx WORD PROCESSING FACTORS xxx
CYBER          0.02      0.02      0.02      0.02      0.02
CREATE         0.00      0.00      0.00      0.00      0.00
HAR 500        0.03      0.03      0.03      0.03      0.03
VAX 11/780     0.15      0.15      0.15      0.15      0.15
```

Table A.8 Example of FACTOR Input File.

Variable	Value	Variable	Value
NUMLOC	3	NUMCPU	5
NUMCLS	6	NUMSIZ	5
NUMWKS	11	NUMST	3
NUMBA	3	NUMAD	3
NUMOTH	1	QTRNUM	1
WPCLAS	6		

Table A.9 RDDIM Namelist Variables.

The RDDIM namelist input can be used to reset any of the model's dimensions. These variables and the defaults are listed in Table A.9. Please see the data dictionary for a description of these varables. The RSNMS namelist statement is used to change any of the computer, class, and size names or IDs listed in Table A.3 or any of the location names. Additionally, the input (INFLID) and output (OTFLID) file unique identifiers may be changed. The default value for the input files is "T". The default value for the output files is "C". Namelist statements must begin in column 2.

An example of the namelist input is:

```
&FLAGNM ECHOFG=.T., WRTOUT=.T., SETDIM=.T., SETNMS=.T. &END
&RDDIM NUMCPU=4, QTRNUM=3 &END
&RSNMS COMPNM(4)='IBM 360 ', COMPID(4)='I' &END
```

In this example, the input files will be echoed (ECHOFG) and the output files created (WRTOUT). RDDIM (SETDIM) and RSNMS (SETNMS) namelist inputs are provided. The number of computers (NUMCPU) is changed to 4 and the quarter number is changed to 3 (QTRNUM). A computer name and ID is changed.

Name	Description
STUDxxy	Student interactive hourly arrival rates and frequency distributions.
ADMNxxy	Faculty/Administration interactive arrival rates and frequency distributions.
OTHRxxy	Other (software development) interactive arrival rates and frequency distributions.
BTCHxxy	Batch hourly arrival rates and frequency distributions.
CNSTxxy	Periodic requirements.

xx - Quarter ID
y - Unique Alpha ID

Table A.10 Workload Output Files.

Outputs

All outputs are controlled by input namelist variables. These output 'flags' must be set to true if output is desired. The FLAGNM namelist input indicates the desired output. Three flags (ECHOFG, TABFG, FRQARV) control printed output. The WRTOUT flag indicates whether output files are created.

Printed Output. The ECHOFG, TABFG, and FRQARV flags indicate whether printed output is desired. If ECHOFG is set to true, all input file records used to calculate the workload are printed. If TABFG is set to true, the FACTOR file values are printed. If FRQARV is set to true, the arrival rates and cumulative frequency distributions for each input location are printed.

Output Files. Five output files are created if the WRTOUT flag is set to true. Each output file name consists

of seven characters. The first four identify the type of location. The next two identify the quarter. The last is an unique identifier provided by the user. These files are described in Table A.10.

Job Setup

The model currently runs on the ASD CYBER computer. The job may be entered through interactive batch or regular batch input. Typical job statements are:

```
MCD,T70,IO100,CM140000.  T123456,MCDERMOTT
COMMENT.  ATTACH INPUT FILES
ATTACH,FACTORT,FACTORT.
ATTACH,CRSWRKT,CRSWRKT.
ATTACH,THSWRKT,THSWRKT.
ATTACH,FACRCHT,FACRCHT.
ATTACH,STUDWPT,STUDWPT.
ATTACH,FACTWPT,FACTWPT.
ATTACH,SFTDEVT,SFTDEVT.
ATTACH,DAYREQT,DAYREQT.
ATTACH,WEKREQT,WEKREQT.
ATTACH,PERREQT,PERREQT.
COMMENT.  GET SPACE FOR OUTPUT FILES
REQUEST,STUDWIC,*PF.
REQUEST,BTCHWIC,*PF.
REQUEST,ADMNWIC,*PF.
REQUEST,OTHRWIC,*PF.
REQUEST,CNSTWIC,*PF.
COMMENT.   ATTACH WORKLOAD MODEL AND EXECUTE
ATTACH,WKLD,WKLD.
LOAD,WKLD.
WKLD.
COMMENT.  CATALOG OUTPUT FILES
CATALOG,STUDWIC,,RP=999.
CATALOG,BTCHWIC,,RP=999.
CATALOG,ADMNWIC,,RP=999.
CATALOG,OTHRWIC,,RP=999.
CATALOG,CNSTWIC,,RP=999.
*EOR
  (Namelist inputs)
```

Error Conditions

Minimum error checking is provided by the model. The model's value is totally reflected by the accuracy of the workload descriptions and the FACTOR file. Careful attention should be paid to these files' creation. The only error conditions detected by the model are: factor file does not match input dimensions; computer ID not found; class ID not found; and size ID not found.

Limitations

The workload model capacity is set by FORTRAN parameter statements as defined in the Workload Model Maintenance Guide. Currently the model can handle up to three input locations for each type, a maximum of eleven weeks to the quarter, and a maximum of six computers, classes, and sizes.

APPENDIX B

WORKLOAD MODEL MAINTENANCE GUIDE

Introduction

The AFIT workload model is a top-down program which creates the output files for subsequent use by the network model. The model is written in FORTRAN Version 5 and is currently running on ASD's CYBER computer in batch mode.

The objective of this guide is to review the documentation standards and provide the general structure of the AFIT workload model. Model modification areas are also briefly covered. Please refer to the Workload Model User's Guide (Appendix A) for a description of all input and output files.

Documentation Standards

The bulk of the detailed documentation on the AFIT workload model resides within the coded modules. A specific header was used for this documentation. This standard calls for the following header information on each module:

```
C  Name:
C  Module Number:
C  Function:
C  Input Parameters:
C  Output Parameters:
C  Common Blocks/Variables Used:
C  Common Blocks/Variables Changed:
C  Modules Called:
C  Calling Modules:
```

Besides the header information, comment statements for each module are included within the body of the module.

Each module's structure followed a specific standard. All DO loops and IF THEN ELSE bodies were indented. FORTRAN statements for each module follow the following order:

SUBROUTINE or FUNCTION statement
PARAMETER statements
Local declarations
COMMON statements
The body
RETURN statement
FORMAT statements
END statement

Also included within the model is a BLOCK DATA module. This module includes every common block and sets all default values except logical print flags.

The data dictionary is included in Appendix C. This dictionary also followed a standard format as follows:

Name: Type:
Common Block:
Description:
Initialized by:
Used by:

The items in the data dictionary include all common variables and names for PARAMETER constants.

General Structure

As previously mentioned, the AFIT workload model has a top-down structure. At the top of this structure is the MAIN module (Figure B.1). Table B.1 lists all of the modules with a short description and the page number of source code in Appendix D. First, the module clears the TOTAL arrays which are used to tabulate the total number of users for each type of input location. MAIN initializes the model by calling CONFIG which reads the namelist inputs. CONFIG calls LDFACT to read the FACTORx file and will call PRTTAB to print the FACTORx file. The modules CALSTD, CALADM, CALOTH, and BATCH are called to process the student, faculty/administrative, software development, and batch

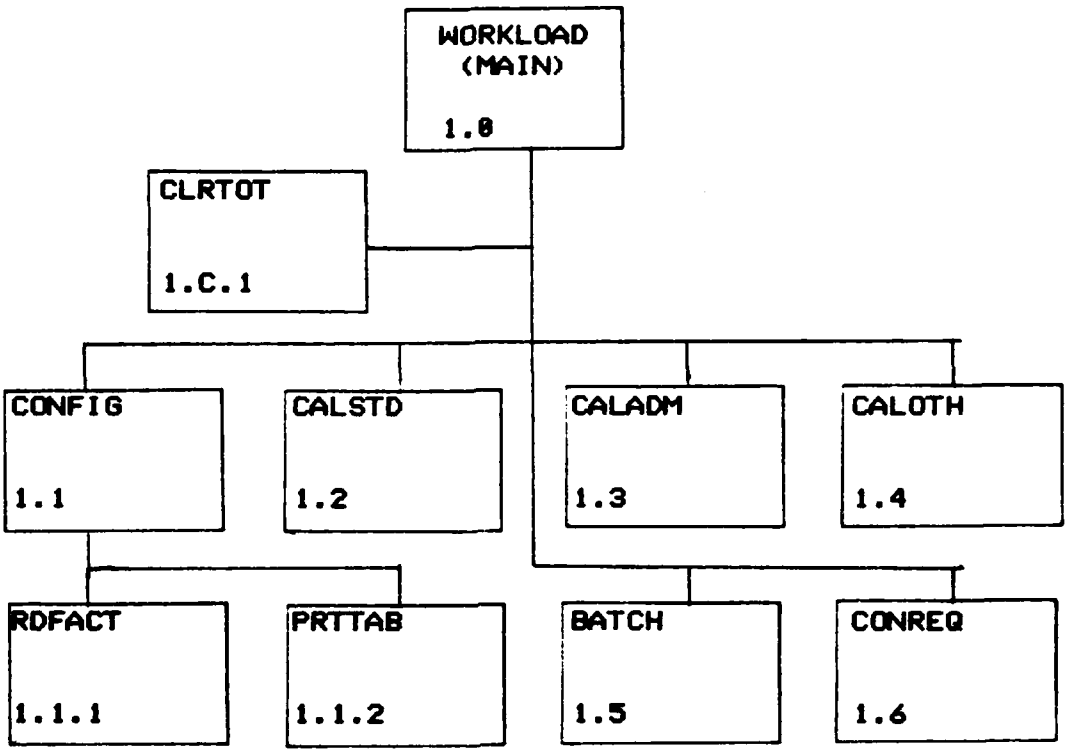


Figure B.1 Workload (MAIN) Hierarchy.

requirements respectfully. Finally CONREQ is called to process the periodic requirements.

Student, faculty/administrative, software development, and batch requirements are all calculated by calling the same set of common modules as shown in Figures B.2 through B.5. Each box represents a module. Within each box is the module's name, file processed in quotation marks, and the module number. Please refer to the user's guide for format and descriptions of the input and output files.

Module REGWRK processes all regular requirements such as student course work (format 1 files). Module WPWORK processes all word processing files (format 2). Module

Module Name	Module Number	Page Number	Description
BATCH	1.5	D-2	Processes all batch locations.
BLOCK	1.8	D-3	BLOCK DATA
CALADM	1.3	D-5	Calculates faculty/administration requirements.
CALOTH	1.4	D-6	Calculates the software development requirements.
CALSTD	1.2	D-7	Calculates the student requirements.
CLRTOT	1.C.1	D-8	Clears the total arrays.
CLSERR	1.C.2	D-9	Processes all close file errors.
CLSFIL	1.C.3	D-9	Closes all files.
CONFIG	1.1	D-10	Reads the input configuration.
CONREQ	1.6	D-11	Processes all constant requirements.
DAYSFT	1.C.4	D-12	Processes all format 3 files.
DISTWP	1.C.5	D-13	Distributes all word processing requirements into the total arrays.
DSTWRK	1.C.6	D-14	Distributes format 1 and format 3 requirements into the total arrays.
FRQHDR	1.C.7	D-15	Prints the arrival and frequency file headers.
OPNERR	1.C.8	D-16	Processes all close file errors.
OPNINP	1.C.9	D-16	Opens all input files.
OPNOUT	1.C.10	D-17	Opens all output files.
PERIOD	1.6.2	D-17	Processes the PERIOD file.
PRTHDR	1.C.11	D-18	Prints the echo input header.
PRTTAB	1.1.2	D-20	Prints the input file tables.
RDERR	1.C.12	D-23	Processes all read errors.
RDFACT	1.1.1	D-23	Reads the input FACTOR file.
RDFMT1	1.C.13	D-25	Reads format 1 files.
RDFMT2	1.C.14	D-26	Reads format 2 files.
RDFMT3	1.C.15	D-26	Reads format 3 files.
RDFMT4	1.C.16	D-28	Reads format 4 files.
REGWRK	1.C.17	D-29	Processes the format 1 files.
SRTWRT	1.6.3	D-30	Sorts and writes the constant requirements.
WEEKLY	1.6.1	D-31	Processes the WEEKLY file.
WPWORK	1.C.18	D-32	Processes the word processing files.
WRTFIL	1.C.19	D-33	Calculates and writes the arrival rates and cumulative frequency distributions output files.
FIGCLS	1.F.1	D-35	Returns class integer.
FIGCPU	1.F.2	D-36	Returns computer integer.
FIGSIZ	1.F.3	D-36	Returns size integer.

Table B.1 Workload Model Modules.

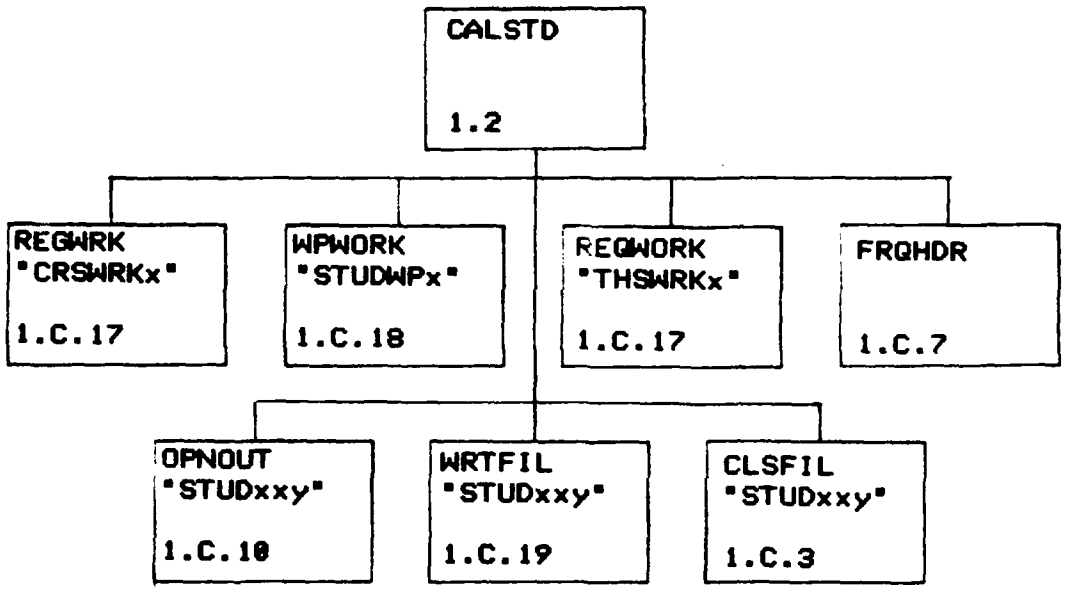


Figure B.2 Student Processing Hierarchy.

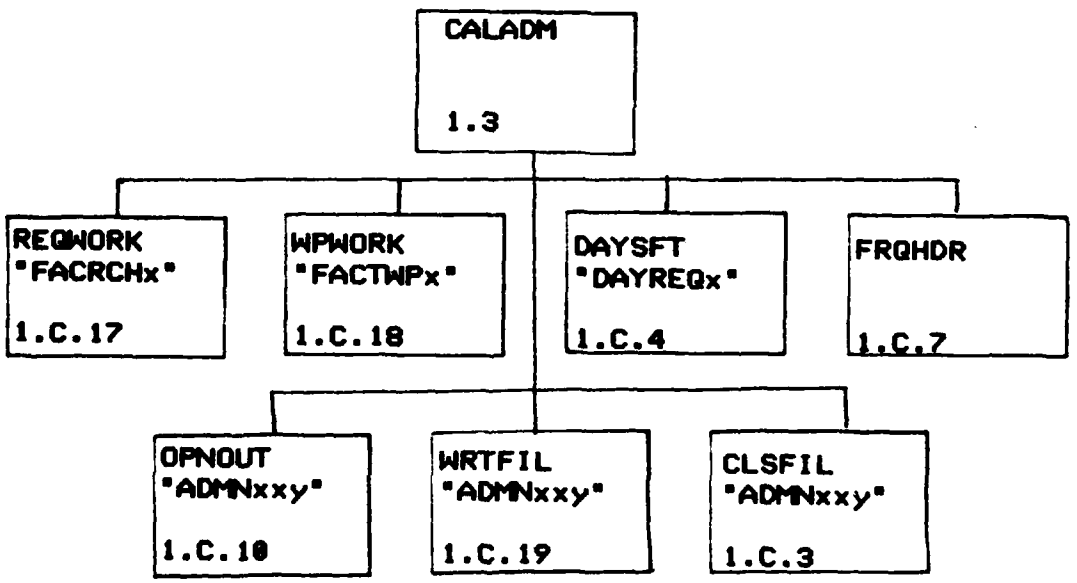


Figure B.3 Faculty/Administration Processing Hierarchy.

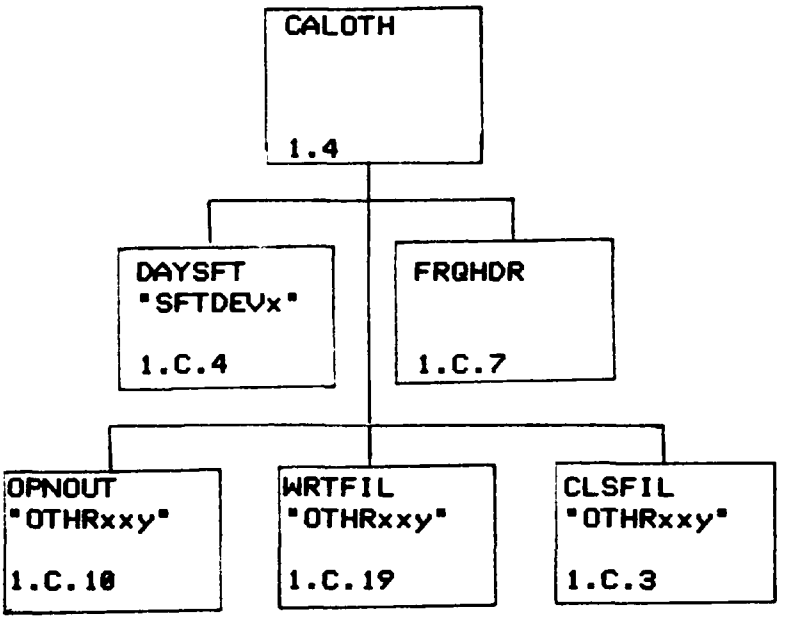


Figure B.4 Software Development Processing Hierarchy.

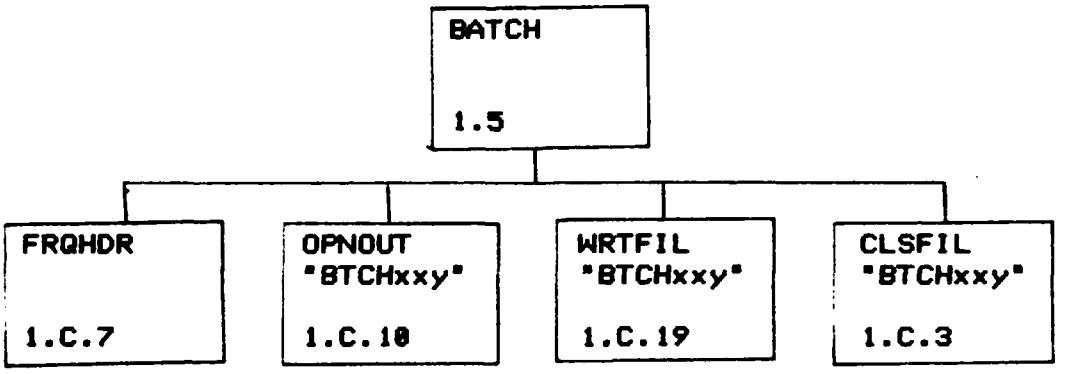


Figure B.5 Batch Processing Hierarchy.

DAYSFT processes both daily requirements and software development requirements (format 3) files. FRQHDR is called to print header information if the arrival rates and cumulative frequency distributions are to be printed. Lastly,

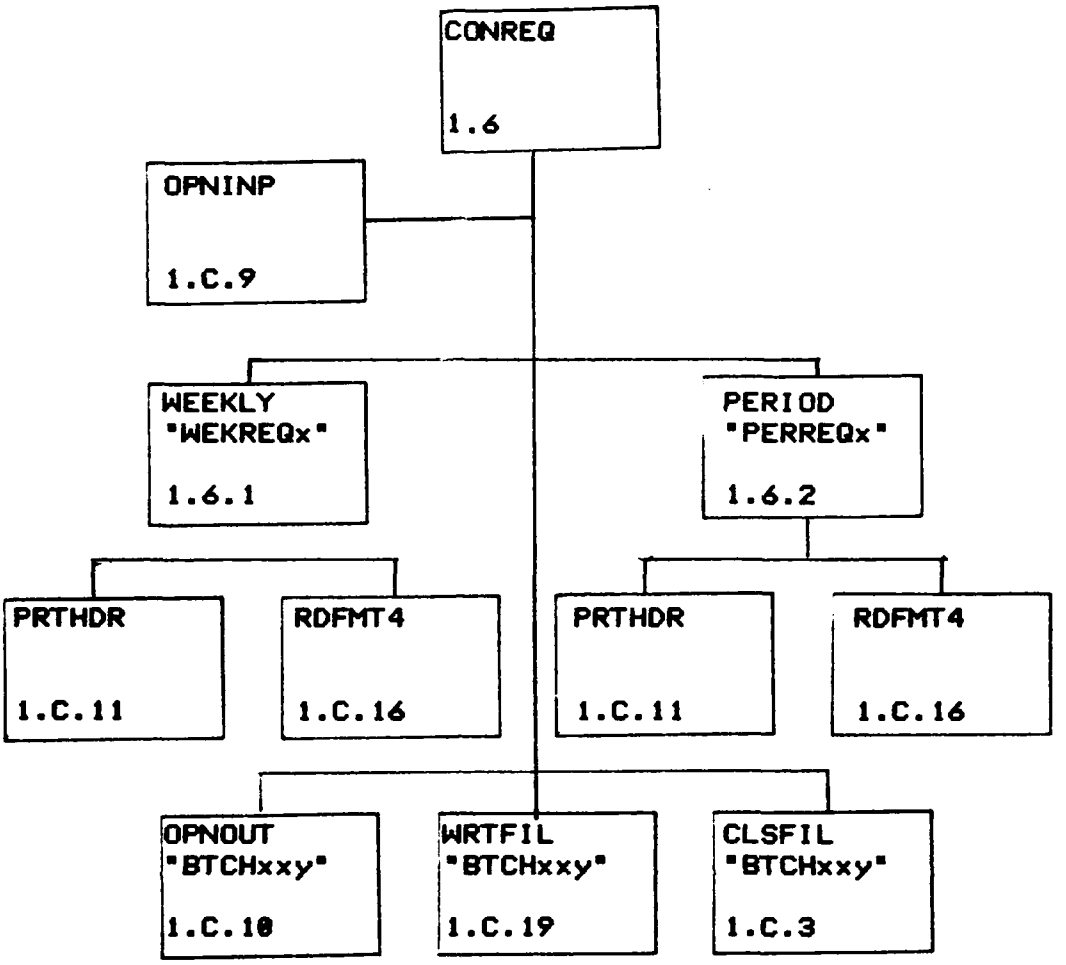


Figure B.6 Periodic Requirements Processing Hierarchy.

OPNOUT, WRTFIL, and CLSFIL are called to open the output files; calculate and output the arrival rates and frequency distributions; and close the output file respectfully. If the output file is not to be created, only WRTFIL is called.

Figure B.6 illustrates the constant requirements (CONREQ) processing. OPNOUT is called to open the input files as they are processed. Modules WEEKLY and PERIOD are called to process the weekly requirements and periodic requirements respectfully (format 4 files). Both of these

modules call PRTHDR if the input files are echoed. RDFMT4 actually reads the files. Finally, OPNOUT, SRTWRT, and CLSFIL are called to open an output file; sort and output the requirements; and close the output file respectfully. If the output file is not created, only SRTWRT is called.

Additions or deletions of input files to the model would occur in these top level modules. For input file addition, the appropriate calls to the lower level modules would need to be added. For deletion, the calls to the lower level modules would be deleted. In both cases, modifications to the FRQHDR and PRTHDR modules would need to be made.

Figures B.7 through B.9 illustrate the different format file processing. REGWRK, WPWORK, and DAYSFT process format 1 through 3 files respectfully. All three modules call OPNINP to open the input files. If the input files are to be echoed, PRTHDR is called to print an appropriate header for each input file. Modules RDFMT1, RDFMT2, and RDFMT3 read the records of each file. Module DSTWRK distributes each record requirements into the 'total' arrays for both format 1 and format 3 files. Module DISTWP is called to distribute word processing requirements (format 2) into the 'total' arrays. Any modifications made to the input file formats will required modifications in these modules.

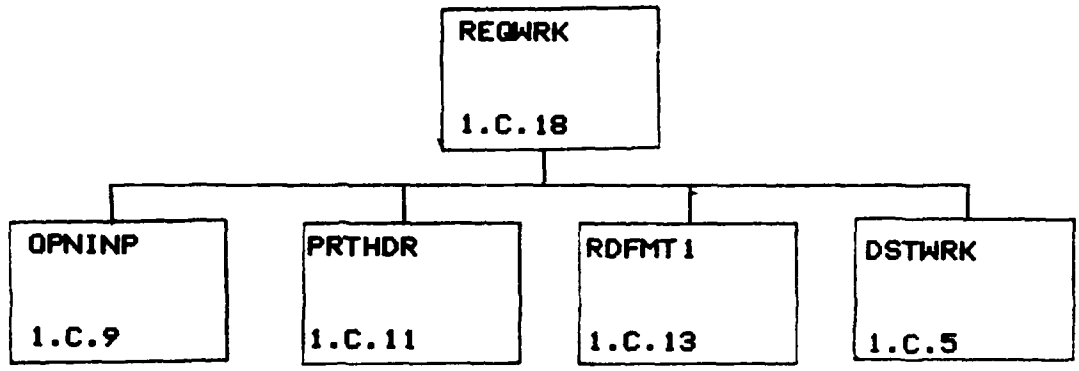


Figure B.7 Format 1 File Processing Hierarchy.

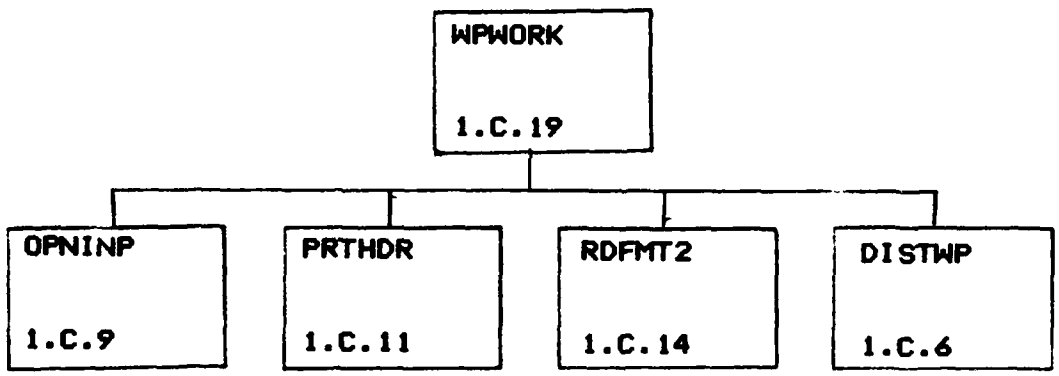


Figure B.8 Format 2 File Processing Hierarchy.

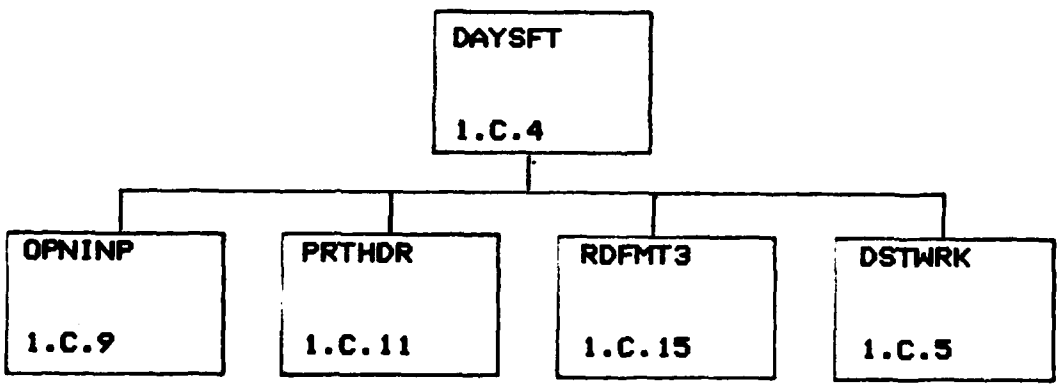


Figure B.9 Format 3 File Processing Hierarchy.

Limitations

The workload model's capacity is set by the FORTRAN parameter statements. To increase the number of input locations, computers, classes, sizes, or number of weeks the appropriate parameter statement must be modified in each module it appears. The data dictionary provides a cross reference for each parameter.

APPENDIX C

WORKLOAD MODEL DATA DICTIONARY

Name: BAFACT(MAXCPU) Type: REAL
Common Block: FACTXX
Description: Identifies whether computer has batch capabilities.
1.0 - Yes
0.0 - No
Initialized by: RDFACT
Used by: PRTTAB

Name: CLASID(MAXCLS) Type: CHAR*1
Common Block: IDS
Description: Workload class ID codes.
Initialized by: BLOCK DATA/CONFIG
Used by: PRTTAB,FIGCLS

Name: CLASNM(MAXCLS) Type: CHAR*8
Common Block: NAMES
Description: Description of class codes.
Initialized by: BLOCK DATA/CONFIG
Used by: PRTTAB,RDFMT3-4,WRTFIL

Name: COMPID(MAXCPU) Type: CHAR*1
Common Block: IDS
Description: Computer ID codes.
Initialized by: BLOCK DATA/CONFIG
Used by: PRTTAB,FIGCPU

Name: COMPNM(MAXCPU) Type: CHAR*8
Common Block: NAMES
Description: Computer descriptions.
Initialized by: BLOCK DATA/CONFIG
Used by: PRTTAB,RDFMT3-4,WRTFIL

Name: DAYSAD(7) Type: REAL
Common Block: DAYSXX
Description: Administation workload distribution over the week.
Initialized by: RDFACT
Used by: CALADM,CALOTH,PRTTAB

Name: DAYSBA(7) Type: REAL
Common Block: DAYSXX
Description: Batch workload distribution over the week.
Initialized by: RDFACT
Used by: BATCH,PRTTAB

Name: DAYSST(7) Type: REAL
Common Block: DAYSXX
Description: Student workload distribution over the week.
Initialized by: RDFACT
Used by: CALSTD,PRTTAB

Name: ECHOFG Type: LOGICAL
Common Block: FLAGS
Description: Flag which indicates whether input files are printed.
Initialized by: CONFIG
Used by: CONFIG,DAYSFT,PERIOD,RDFMT1-4,REGWRK,WEEKLY,WPWORK

Name: FRQARV Type: LOGICAL
Common Block: FLAGS
Description: Flag which indicates whether arrival rates and cummulative frequency distributions are printed for each input location.
Initialized by: CONFIG
Used by: BATCH,CALADM,CALOTH,CALSTD,CONFIG,CONREQ,SRTWRT, WRTFIL

Name: HOURAD(24) Type: REAL
Common Block: HOURXX
Description: Adminstrative/faculity workload distribution over the day.
Initialized by: RDFACT
Used by: CALADM,CALOTH,PRTTAB

Name: HOURBA(24) Type: REAL
Common Block: HOURXX
Description: Batch workload distribution over the day.
Initialized by: RDFACT
Used by: BATCH,PRTTAB

Name: HOURST(24) Type: REAL
Common Block: HOURXX
Description: Student workload distribution over the day.
Initialized by: RDFACT
Used by: CALSTD,PRTTAB

Name: INFLID(10) Type: CHAR*1
Common Block: FILIDS
Description: Unique input file suffixes.
Initialized by: BLOCK DATA/CONFIG
Used by: CALADM,CALOTH,CALSTD,CONFIG,CONREQ

Name: LOCNMS(MAXLOC) Type: CHAR*8
Common Block: NAMES
Description: Location names.
Initialized by: BLOCK DATA/CONFIG
Used by: RDFMT1-3,WRTFIL

Name: MAXCPU Type: PARAMETER
Common Block: N/A
Description: Maximum number of computers allowed.
Used by: MAIN,BATCH,BLOCK DATA,CALADM,CALOTH,CALSTD,CLRTOT,
CONFIG,DSTWRK,DISTWP,PERIOD,PRTTAB,RDFACT,RDFMT1-4,
SRTWRT,WEEKLY,WRTFIL,FIGCLS,FIGCPU,FIGSIZ

Name: MAXCLS Type: PARAMETER
Common Block: N/A
Description: Maximum number of workload classes allowed.
Used by: MAIN,BATCH,BLOCK DATA,CALADM,CALOTH,CALSTD,CLRTOT,
CONFIG,DSTWRK,DISTWP,PERIOD,PRTTAB,RDFACT,RDFMT1-4,
SRTWRT,WEEKLY,WRTFIL,FIGCLS,FIGCPU,FIGSIZ

Name: MAXLOC Type: PARAMETER
Common Block: N/A
Description: Maximum number of different locations allowed.
Used by: MAIN,BATCH,BLOCK DATA,CALADM,CALOTH,CALSTD,CLRTOT,
CONFIG,DSTWRK,DISTWP,PERIOD,PRTTAB,RDFMT1-4,SRTWRT,
WEEKLY,WRTFIL,FIGCLS,FIGCPU,FIGSIZ

Name: MAXSIZ Type: PARAMETER
Common Block: N/A
Description: Maximum number of workload memory sizes allowed.
Used by: MAIN,BATCH,BLOCK DATA,CALADM,CALOTH,CALSTD,CLRTOT,
CONFIG,DSTWRK,DISTWP,PERIOD,PRTTAB,RDFACT,RDFMT1-4,
SRTWRT,WEEKLY,WRTFIL,FIGCLS,FIGCPU,FIGSIZ

Name: MAXWKS Type: PARAMETER
Common Block: N/A
Description: Maximum number of weeks allowed.
Used by: MAIN,BATCH,BLOCK DATA,CALADM,CALOTH,CALSTD,CLRTOT,
CONFIG,DSTWRK,DISTWP,PERIOD,RDFMT1-4,REGWRK,SRTWRT,
WEEKLY,WPWORK,WRTFIL,FIGCLS,FIGCPU,FIGSIZ

Name: NCRDR Type: INTEGER
Common Block: UNITS
Description: Input (card) input number.
Initialized by: BLOCK DATA
Used by: CONFIG

Name: NINPUT Type: INTEGER
Common Block: UNITS
Description: Input files unit number.
Initialized by: BLOCK DATA
Used by: CONREQ,DAYSFT,OPNINP,RDFACT,RDFMT1-4,REGWRK,WPWORK

Name: NOUT Type: INTEGER
Common Block: UNITS
Description: Output files unit number.
Initialized by: BLOCK DATA
Used by: CLSFIL,OPNOUT,SRTWRT,WRTFIL

Name: NPRNT Type: INTEGER
Common Block: UNITS
Description: Printer unit number.
Initialized by: BLOCK DATA
Used by: CLSERR,CLSFIL,FRQHDR,OPNERR,PRTHDR,PRTTAB,RDERR,
RDFMT1-4,SRTWRT,WRTFIL,FIGCLS,FIGCPU,FIGSIZ

Name: NUMAD Type: INTEGER
Common Block: DIMEN
Description: Number of adminstative/faculity locations.
Initialized by: BLOCK DATA/CONFIG
Used by: CALADM

Name: NUMBA Type: INTEGER
Common Block: DIMEN
Description: Number of batch locations.
Initialized by: BLOCK DATA/CONFIG
Used by: BATCH

Name: NUMCLS Type: INTEGER
Common Block: DIMEN
Description: Number of classes.
Initialized by: BLOCK DATA/CONFIG
Used by: CLRTOT,PRTTAB,RDFACT,WRTFIL,FIGCLS

Name: NUMCPU Type: INTEGER
Common Block: DIMEN
Description: Number of computers.
Initialized by: BLOCK DATA/CONFIG
Used by: CLRTOT,DISTWP,PRTTAB,RDFACT,WRTFIL,FIGCPU

Name: NUMLOC Type: INTEGER
Common Block: DIMEN
Description: Number of locations.
Initialized by: BLOCK DATA/CONFIG
Used by: CLRTOT

Name: NUMOTH Type: INTEGER
Common Block: DIMEN
Description: Number of other (software development)
locations.
Initialized by: BLOCK DATA/CONFIG
Used by: CALOTH

Name: NUMSIZ Type: INTEGER
Common Block: DIMEN
Description: Number of workload sizes.
Initialized by: BLOCK DATA/CONFIG
Used by: CLRTOT,DISTWP,PRTTAB,RDFACT,WRTFIL,FIGSIZ

Name: NUMST Type: INTEGER
Common Block: DIMEN
Description: Number of student locations.
Initialized by: BLOCK DATA/CONFIG
Used by: CALSTD

Name: NUMWKS Type: INTEGER
Common Block: DIMEN
Description: Number of weeks.
Initialized by: BLOCK DATA/CONFIG
Used by: CLRTOT,DAYSFT,REGWRK,WEEKLY,WPWORK,WRTFIL

Name: OTFLID Type: CHAR*1
Common Block: FILIDS
Description: Unique output file suffix.
Initialized by: BLOCK DATA/CONFIG
Used by: BATCH,CALADM,CALOTH,CALSTD,CONFIG,CONREQ

Name: PERNMS(10) Type: CHAR*8
Common Block: NAMES
Description: Periodic location names (corresponds to Network Simulation Model).
Initialized by: BLOCK DATA/CONFIG
Used by: RDFMT4

Name: PCFACT(MAXCLS,MAXSIZ) Type: REAL
Common Block: FACTXX
Description: Factor which indicates whether workload for each class and size is capable of being performed by a personnal computer.
Initialized by: RDFACT
Used by: PRTTAB,DSTWRK

Name: QTRNMS(5) Type: CHAR*2
Common Block: QTRDEF
Description: Quarter names (WI,SP,SU,FA,AL).
Initialized by: BLOCK DATA/CONFIG
Used by: BATCH,CALADM,CALOTH,CALSTD,CONREQ,PRTHDR,RDFMT2-4

Name: QTRNUM Type: INTEGER
Common Block: DIMEN
Description: Quarter number of the run (1-WI, 2-SP, 3-WU, 4-FA).
Initialized by: BLOCK DATA/CONFIG
Used by: BATCH,CALADM,CALOTH,CALSTD,CONREQ,PRTHDR,RDFMT1-4

Name: RUNFACT(MAXCLS,MAXSIZ) Type: REAL
Common Block: FACTXX
Description: Number of batch runs required per user for each class and size.
Initialized by: RDFACT
Used by: PRTTAB,WRTFIL

Name: SESFACT(MAXCLS,MAXSIZ) Type: REAL
Common Block: FACTXX
Description: Number of interactive sessions required per user for each class and size.
Initialized by: RDFACT
Used by: PRTTAB,WRTFIL

Name: SIZIDS(MAXSIZ) Type: CHAR*1
Common Block: IDS
Description: Memory size codes.
Initialized by: BLOCK DATA/CONFIG
Used by: PRTTAB,FIGSIZ

Name: SIZNMS Type: CHAR*8
Common Block: NAMES
Description: Memory size names.
Initialized by: BLOCK DATA/CONFIG
Used by: PRTTAB,RDFMT3-4

Name: TABFG Type: LOGICAL
Common Block: FLAGS
Description: Flag which indicates whether input parameters/variables are to be printed.
Initialized by: CONFIG
Used by: CONFIG

Name: TOTBAT(MAXLOC,MAXWKS,MAXCPU,MAXCLS, MAXSIZ) Type: REAL
Common Block: TOTALS
Description: Array used to tabulate the number of users of batch input locations by location, week, computer, class size.
Description: x
Initialized by: MAIN
Used by: MAIN,BATCH

Name: TOTINT(MAXLOC,MAXWKS,MAXCPU,MAXCLS, MAXSIZ) Type: REAL
Common Block: TOTALS
Description: Array used to tabulate the number of users of interactive input locations by location, week, computer, class and size.
Description: x
Initialized by: MAIN
Used by: MAIN,CALADM,CALOTH,CALSTD,DISTWP,PERIOD,SRTWRT, WEEKLY

Name: TOTPC(MAXLOC,MAXWKS,MAXCPU,MAXCLS, MAXSIZ) Type: REAL
Common Block: TOTALS
Description: Array used to tabulate the number of users of personal computers by location, week, computer, class and size.
Initialized by: MAIN
Used by: MAIN,DISTWP

Name: TYPLOC(MAXLOC) Type: CHAR*8
Common Block: NAMES
Description: Types of input locations.
Initialized by: BLOCK DATA/CONFIG
Used by: PRTTAB

Name: WPCLAS Type: INTEGER
Common Block: DIMEN
Description: Word processing class number.
Initialized by: BLOCK DATA/CONFIG
Used by: DISTWP

Name: WPFACT(MAXCPU,MAXSIZ) Type: REAL
Common Block: FACTXX
Description: Cummulative frequency distribution used to distribute word processing load by computer and size.
Initialized by: RDFACT
Used by: DISTWP,PRTTAB

Name: WRTOUT Type: LOGICAL
Common Block: FLAGS
Description: Flag which indicates whether output files are created.
Initialized by: CONFIG
Used by: BATCH,CALADM,CALOTH,CALSTD,CONFIG,CONREQ,SRTWRT, WRTFIL

APPENDIX D

WORKLOAD MODEL SOURCE CODE

```
C*************************************************************************
C                                                                        *
C  Name: WORKLOAD (MAIN)                                                 *
C  Module Number: 1.0                                                    *
C  Function: WORKLOAD reads input files describing the AFIT work-        *
C            load and creates output files for use by the system         *
C            network model.                                              *
C  Input Parameters: none                                                *
C  Output Parameters: none                                               *
C  Common Blocks/Variables Used: TOTALS/TOTINT,TOTBAT,TOTPC              *
C  Common Blocks/Variables Changed: TOTALS/TOTINT,TOTBAT,TOTPC           *
C  Modules Called: CLRTOT,CONFIG,CALADM,CALSTD,CALOTH,BATCH,CONREQ       *
C  Calling Modules: N/A                                                  *
C                                                                        *
C*************************************************************************
C
      PROGRAM MAIN(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      PARAMETER (MAXLOC=3,MAXWKS=11,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
C**** COMMON BLOCKS
      COMMON/TOTALS/ TOTINT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTBAT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTPC(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ)
C
C**** READ IN INPUT PARAMETERS AND FACTOR FILE
      CALL CONFIG
C
C**** CLEAR THE TOTAL ARRAYS
      CALL CLRTOT(TOTINT)
      CALL CLRTOT(TOTBAT)
      CALL CLRTOT(TOTPC)
C
C**** CALCULATE THE STUDENT WORKLOAD
      CALL CALSTD
      CALL CLRTOT(TOTINT)
      CALL CLRTOT(TOTPC)
C
C**** CALCULATE THE FACULITY/ADMIN WORKLOAD
      CALL CALADM
      CALL CLRTOT(TOTINT)
      CALL CLRTOT(TOTPC)
C
C**** CALCULATE THE SOFTWARE DEVELOPMENT WORKLOAD
      CALL CALOTH
C
C**** CALCULATE THE BATCH WORKLOAD
      CALL BATCH
C
C**** READ AND FORMAT THE PERIODIC REQUIREMENTS
      CALL CONREQ
      END
C*************************************************************************
C                                                                        *
C  Name: BATCH                                                           *
C  Module Number: 1.5                                                    *
```

```
C  Function: Calls modules to create the batch arrival and frequency*
C            files.                                                  *
C  Input Parameters: none                                            *
C  Output Parameters: none                                           *
C  Common Blocks/Variables Used: DAYSXX/DAYSBA                       *
C       DIMEN/QTRNUM,NUMBA                                           *
C       FILIDS/OTFLID                                                *
C       FLAGS/FRQARV,WRTOUT                                          *
C       HOURXX/HOURBA                                                *
C       QTRDEF/QTRNMS                                                *
C       TOTALS/TOTBAT                                                *
C  Common Blocks/Variables Changed: none                             *
C  Modules Called: FRQHDR,OPNFIL,WRTFIL,CLSFIL                       *
C  Calling Modules: MAIN                                             *
C                                                                    *
C********************************************************************
C
      SUBROUTINE BATCH
      PARAMETER (MAXLOC=3,MAXWKS=11,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      CHARACTER OUTFIL*7
C
C**** COMMON BLOCKS
      COMMON/DAYSXX/ DAYSST(7),DAYSAD(7),DAYSBA(7)
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +    NUMAD,NUMOTH,QTRNUM,WPCLAS
      CHARACTER*1 INFLID,OTFLID
      COMMON/FILIDS/ INFLID(10),OTFLID
      LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/HOURXX/ HOURST(24),HOURAD(24),HOURBA(24)
      CHARACTER*2 QTRNMS
      COMMON/QTRDEF/ QTRNMS(5)
      COMMON/TOTALS/ TOTINT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTBAT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTPC(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ)
C
C**** OPEN FILE AND WRITE HEADER IF FRQARV
      OUTFIL='BTCH'//QTRNMS(QTRNUM)(:2)//OTFLID(:1)
      IF (FRQARV) CALL FRQHDR(OUTFIL,4)
C
C**** IF WRTOUT THEN OPEN FILE - CALL WRTFIL
      IF (WRTOUT) THEN
          CALL OPNOUT(OUTFIL,'U')
          CALL WRTFIL(TOTBAT,NUMBA,DAYSBA,HOURBA,3)
          CALL CLSFIL(OUTFIL)
      ELSE
          CALL WRTFIL(TOTBAT,NUMBA,DAYSBA,HOURBA,3)
      ENDIF
      RETURN
      END
C********************************************************************
C                                                                    *
C  Name: BLOCK DATA                                                  *
```

```
C  Module Number: 1.B                                                    *
C  Function: Defines all common blocks and set variable defaults.        *
C  Input Parameters: N/A                                                 *
C  Output Parameters: N/A                                                *
C  Common Blocks/Variables Used: All                                     *
C  Common Blocks/Variables Changed: See Data Statements.                 *
C  Modules Called: N/A                                                   *
C  Calling Modules: N/A                                                  *
C                                                                        *
C************************************************************************
C
      BLOCK DATA
      PARAMETER (MAXLOC=3,MAXWKS=11,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      CHARACTER*1 COMPID,CLASID,SIZIDS,INFLID,OTFLID
      CHARACTER*8 CLASNM,COMPNM,LOCNMS,SIZNMS,TYPLOC,PERNMS
      CHARACTER*2 QTRNMS
C
      INTEGER QTRNUM,WPCLAS
C
      LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
C
      COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,
     +    NUMST,NUMBA,NUMAD,NUMOTH,QTRNUM,WPCLAS
      COMMON/DAYSXX/ DAYSST(7),DAYSAD(7),DAYSBA(7)
      COMMON/HOURXX/ HOURST(24),HOURAD(24),HOURBA(24)
      COMMON/FACTXX/ PCFACT(MAXCLS,MAXSIZ),SESFACT(MAXCLS,MAXSIZ),
     +  RUNFACT(MAXCLS,MAXSIZ),BAFACT(MAXCPU),WPFACT(MAXCPU,MAXSIZ)
      COMMON/FILIDS/ INFLID(10),OTFLID
      COMMON/QTRDEF/ QTRNMS(5)
      COMMON/TOTALS/ TOTINT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTBAT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTPC(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ)
      COMMON/IDS/ COMPID(MAXCPU),CLASID(MAXCLS),SIZIDS(MAXSIZ)
      COMMON/NAMES/ CLASNM(MAXCLS),COMPNM(MAXCPU),LOCNMS(MAXLOC),
     +    SIZNMS(MAXSIZ),TYPLOC(MAXLOC),PERNMS(10)
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
C**** DATA STATEMENTS
C
      DATA NUMLOC,NUMCPU,NUMCLS,NUMSIZ/3,5,6,5/
      DATA NUMWKS,NUMST,NUMBA,NUMAD,NUMOTH/11,3,3,3,1/
      DATA QTRNUM,WPCLAS/1,6/
      DATA INFLID,OTFLID/10*'T','C'/
      DATA CLASID/'A','B','C','D','E','F'/
      DATA CLASNM/'BASIC LG','DBMS    ','SPSS    ','SLAM    ',
     +    'CANNED  ','WORDPROC'/
      DATA COMPID/'C','R','H','V','A',' '/
      DATA COMPNM/'CYBERS  ','CREATE  ','HAR 500 ','SSC     ',
     +    'ANY     ','UNDEF   '/
      DATA NCRDR,NPRNT,NINPUT,NOUT/5,6,7,8/
      DATA LOCNMS/'BLD 125 ','BLD 640 ','BLD 641'/
      DATA QTRNMS/'WI','SP','SU','FA','AL'/
      DATA SIZIDS/'1','2','3','4','5','6'/
```

```
      DATA SIZNMS/' < 20K  ','20 - 40K','40-100K ','100-200K',
     +    '200-300K','OVR 300K'/
      DATA TYPLOC/'STUDENT ','BATCH   ','FAC/ADMN'/
      DATA PERNMS/'STUD125 ','STUD640 ','STUD641 ','ADMN125 ',
     +    'ADMN640 ','ADMN641 ','SOFT125 ','BATCH125','BATCH640',
     +    'BATCH641'/
      END
C*************************************************************************
C                                                                        *
C  Name: CALADM                                                          *
C  Module Number: 1.3                                                    *
C  Function: Performs all calls to subroutines to read input files       *
C            for faculity/admin workload and create the arrival          *
C            and frequency files.                                        *
C  Input Parameters: none                                                *
C  Output Parameters: none                                               *
C  Common Blocks/Variables Used: DAYSXX/DAYSAD                           *
C      DIMEN/QTRNUM,NUMAD                                                *
C      FILIDS/INFLID,OTFLID                                              *
C      FLAGS/FRQARV,WRTOUT                                               *
C      HOURXX/HOURAD                                                     *
C      QTRDEF/QTRNMS                                                     *
C      TOTALS/TOTINT                                                     *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: REGWRK,DAYSFT,WPWORK,WRTFIL,FRQHDR,OPNFIL,CLSFIL      *
C  Calling Modules: MAIN                                                 *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE CALADM
      PARAMETER (MAXLOC=3,MAXWKS=11,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      CHARACTER FILENM*7,OUTFIL*7
C
C**** COMMON BLOCKS
      COMMON/DAYSXX/ DAYSST(7),DAYSAD(7),DAYSBA(7)
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +    NUMAD,NUMOTH,QTRNUM,WPCLAS
      CHARACTER*1 INFLID,OTFLID
      COMMON/FILIDS/ INFLID(10),OTFLID
      LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/HOURXX/ HOURST(24),HOURAD(24),HOURBA(24)
      CHARACTER*2 QTRNMS
      COMMON/QTRDEF/ QTRNMS(5)
      COMMON/TOTALS/ TOTINT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTBAT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTPC(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ)
C
C**** READ AND PROCESS FACRCH FILE
      FILENM='FACRCH'//INFLID(4)
      CALL REGWRK(FILENM,4)
C
C**** READ AND PROCESS FACULITY WORD PROCESSING REQUIREMENTS
```

```
      FILENM='FACTWP'//INFLID(9)
      CALL WPWORK(FILENM,9)
C
C**** READ AND PROCESS DAILY REQUIREMENTS
      FILENM='DAYREQ'//INFLID(6)
      CALL DAYSFT(FILENM,6)
C
C**** OPEN AND WRITE OUT ADMIN ARRIVAL FILE
      OUTFIL='ADMN'//QTRNMS(QTRNUM)(:2)//OTFLID(:1)
      IF (FRQARV) CALL FRQHDR(OUTFIL,2)
      IF (WRTOUT) THEN
          CALL OPNOUT(OUTFIL,'U')
          CALL WRTFIL(TOTINT,NUMAD,DAYSAD,HOURAD,1)
          CALL CLSFIL(OUTFIL)
      ELSE
          CALL WRTFIL(TOTINT,NUMAD,DAYSAD,HOURAD,1)
      ENDIF
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: CALOTH                                                          *
C  Module Number: 1.                                                     *
C  Function: Performs all calls to subroutines to read input files       *
C            for software development workload and create the out-       *
C            put arrival and frequency file.                             *
C  Input Parameters: none                                                *
C  Output Parameters: none                                               *
C  Common Blocks/Variables Used: DAYSXX/DAYSAD                           *
C      DIMEN/QTRNUM,NUMOTH                                               *
C      FILIDS/INFLID,OTFLID                                              *
C      FLAGS/FRQARV,WRTOUT                                               *
C      HOURXX/HOURAD                                                     *
C      QTRDEF/QTRNMS                                                     *
C      TOTALS/TOTINT                                                     *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: DAYSFT, WRTFIL,FRQHDR,OPNFIL,CLSFIL                   *
C  Calling Modules: MAIN                                                 *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE CALOTH
      PARAMETER (MAXLOC=3,MAXWKS=11,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      CHARACTER FILENM*7,OUTFIL*7
C
C**** COMMON BLOCKS
      COMMON/DAYSXX/ DAYSST(7),DAYSAD(7),DAYSBA(7)
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +    NUMAD,NUMOTH,QTRNUM,WPCLAS
      CHARACTER*1 INFLID,OTFLID
      COMMON/FILIDS/ INFLID(10),OTFLID
      LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
```

```
      COMMON/HOURXX/ HOURST(24),HOURAD(24),HOURBA(24)
      CHARACTER*2 QTRNMS
      COMMON/QTRDEF/ QTRNMS(5)
      COMMON/TOTALS/ TOTINT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTBAT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTPC(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ)
C
C**** READ AND PROCESS SFTDEV FILE
      FILENM='SFTDEV'//INFLID(5)
      CALL DAYSFT(FILENM,5)
C
C**** OPEN AND WRITE OUT OTHR FILE
      OUTFIL='OTHR'//QTRNMS(QTRNUM)(:2)//OTFLID(:1)
      IF (FRQARV) CALL FRQHDR(OUTFIL,3)
      IF (WRTOUT) THEN
          CALL OPNOUT(OUTFIL,'U')
          CALL WRTFIL(TOTINT,NUMOTH,DAYSAD,HOURAD,1)
          CALL CLSFIL(OUTFIL)
      ELSE
          CALL WRTFIL(TOTINT,NUMOTH,DAYSAD,HOURAD,1)
      ENDIF
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: CALSTD                                                          *
C  Module Number: 1.2                                                    *
C  Function: Performs all calls to subroutines to read input files       *
C            to determine student workload and create the arrival        *
C            and frequency output files.                                 *
C  Input Parameters: none                                                *
C  Output Parameters: none                                               *
C  Common Blocks/Variables Used: DAYSXX/DAYSST                           *
C      DIMEN/QTRNUM,NUMST                                                *
C      FILIDS/INFLID,OTFLID                                              *
C      FLAGS/FRQARV,WRTOUT                                               *
C      HOURXX/HOURST                                                     *
C      QTRDEF/QTRNMS                                                     *
C      TOTALS/TOTINT                                                     *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: REGWRK,WPWORK,WRTFIL,FRQHDR,OPNFIL,CLSFIL             *
C  Calling Modules: MAIN                                                 *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE CALSTD
      PARAMETER (MAXLOC=3,MAXWKS=11,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      CHARACTER FILENM*7,OUTFIL*7
C
C**** COMMON BLOCKS
      COMMON/DAYSXX/ DAYSST(7),DAYSAD(7),DAYSBA(7)
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +    NUMAD,NUMOTH,QTRNUM,WPCLAS
```

```
      CHARACTER*1 INFLID,OTFLID
      COMMON/FILIDS/ INFLID(10),OTFLID
      LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/HOURXX/ HOURST(24),HOURAD(24),HOURBA(24)
      CHARACTER*2 QTRNMS
      COMMON/QTRDEF/ QTRNMS(5)
      COMMON/TOTALS/ TOTINT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTBAT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTPC(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ)
C
C**** READ AND PROCESS STUDENT COURSE WORK
      FILENM='CRSWRK'//INFLID(1)
      CALL REGWRK(FILENM,1)
C
C**** READ AND PROCESS STUDENT WORD PROCESSING
      FILENM='STUDWP'//INFLID(2)
      CALL WPWORK(FILENM,2)
C
C**** READ AND PROCESS STUDENT THESIS WORK
      FILENM='THSWRK'//INFLID(3)
      CALL REGWRK(FILENM,3)
C
C**** OPEN AND WRITE STUDENT ARRIVAL RATES
      OUTFIL='STUD'//QTRNMS(QTRNUM)(:2)//OTFLID(:1)
      IF (FRQARV) CALL FRQHDR(OUTFIL,1)
      IF (WRTOUT) THEN
          CALL OPNOUT(OUTFIL,'U')
          CALL WRTFIL(TOTINT,NUMST,DAYSST,HOURST,1)
          CALL CLSFIL(OUTFIL)
      ELSE
          CALL WRTFIL(TOTINT,NUMST,DAYSST,HOURST,1)
      ENDIF
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: CLRTOT                                                          *
C  Module Number: 1.C.1                                                  *
C  Function: Clears a TOTAL array by setting all values to 0.0.          *
C  Input Parameters: TOTAL                                               *
C  Output Parameters: TOTAL                                              *
C  Common Blocks/Variables Used: DIMEN/NUMLOC,NUMWKS,NUMCPU,NUMCLS,      *
C      NUMSIZ                                                            *
C  Common Blocks/Variables Changed: none (directly)                      *
C  Modules Called: none                                                  *
C  Calling Modules: MAIN                                                 *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE CLRTOT(TOTAL)
      PARAMETER (MAXLOC=3,MAXWKS=11,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      REAL TOTAL(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ)
C
```

```
C**** COMMON BLOCKS
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +    NUMAD,NUMOTH,QTRNUM,WPCLAS
C
C**** CLEAR TOTAL ARRAY
      DO 500  I=1,NUMLOC
          DO 400  J=1,NUMWKS
              DO 300 K=1,NUMCPU
                  DO 200  L=1,NUMCLS
                      DO 100  M=1,NUMSIZ
                          TOTAL(I,J,K,L,M)=0.0
  100                 CONTINUE
  200             CONTINUE
  300         CONTINUE
  400     CONTINUE
  500 CONTINUE
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: CLSERR                                                          *
C  Module Number: 1.C.2                                                  *
C  Function: Prints filename when close error occurs and stops           *
C            execution.                                                  *
C  Input Parameters: FILENM                                              *
C  Output Parameters: none                                               *
C  Common Blocks/Variables Used: UNITS/NPRNT                             *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: none                                                  *
C  Calling Modules: CLSFIL,DAYSFT,RDFACT,REGWRK,WPWORK                   *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE CLSERR(FILENM)
      CHARACTER FILENM*7
C
C**** COMMON BLOCKS
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
      WRITE(NPRNT,10) FILENM
      STOP
   10 FORMAT(' ','FILE ',A7,' CLOSE ERROR')
      END
C*************************************************************************
C                                                                        *
C  Name: CLSFIL                                                          *
C  Module Number: 1.C.3                                                  *
C  Function: Closes the arrival and frequency files named OUTFIL         *
C  Input Parameters: OUTFIL                                              *
C  Output Parameters: none                                               *
C  Common Blocks/Variables Used: UNITS/NOUT,NPRNT                        *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: CLSERR                                                *
C  Calling Modules: CALADM,BATCH,CALOTH,CALSTD,CONREQ                     *
```

```
C                                                                       *
C***********************************************************************
C
      SUBROUTINE CLSFIL(OUTFIL)
      CHARACTER OUTFIL*7
C
C**** COMMON BLOCKS
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
      ENDFILE(NOUT,ERR=100)
      CLOSE(NOUT,ERR=200)
      RETURN
  100 WRITE(NPRNT,110) OUTFIL
      STOP
  110 FORMAT(' FILE = ',A7,' ENDFILE ERROR')
  200 CALL CLSERR(OUTFIL)
      END
C***********************************************************************
C                                                                       *
C  Name: CONFIG                                                         *
C  Module Number:  1.1                                                  *
C  Function: Reads the input parameters                                 *
C  Input Parameters: none                                               *
C  Output Parameters: none                                              *
C  Common Blocks/Variables Used: FILIDS/INFILID,OTFILID                 *
C       FLAGS/ECHOFG,TABFG,FRQARV,WRTOUT                                *
C       UNITS/NCRDR                                                     *
C  Common Blocks/Variables Changed: May change common blocks DIMEN      *
C       FILIDS, FLAGS, IDS, and NAMES default values.                   *
C  Modules Called: RDFACT, PRTTAB                                       *
C  Calling Modules: MAIN                                                *
C                                                                       *
C***********************************************************************
C
      SUBROUTINE CONFIG
      PARAMETER (MAXLOC=3,MAXMKS=11,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      LOGICAL SETNMS,SETDIM
C
C**** COMMON BLOCKS
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMMKS,NUMST,NUMBA,
     +     NUMAD,NUMOTH,QTRNUM,WPCLAS
      CHARACTER*1 INFLID,OTFLID
      COMMON/FILIDS/ INFLID(10),OTFLID
      LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
      CHARACTER*1 COMPID,CLASID,SIZIDS
      COMMON/IDS/ COMPID(MAXCPU),CLASID(MAXCLS),SIZIDS(MAXSIZ)
      CHARACTER*8 CLASNM,COMPNM,LOCNMS,SIZNMS,TYPLOC,PERNMS
      COMMON/NAMES/ CLASNM(MAXCLS),COMPNM(MAXCPU),LOCNMS(MAXLOC),
     +     SIZNMS(MAXSIZ),TYPLOC(MAXLOC),PERNMS(10)
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
C**** NAMELIST DECLARATIONS
```

```
      NAMELIST/FLAGNM/ ECHOFG,TABFG,FRQARV,WRTOUT,SETDIM,SETNMS
      NAMELIST/RDDIM/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +  NUMAD,NUMOTH,QTRNUM,WPCLAS
      NAMELIST/RSNMS/ CLASNM,COMPNM,LOCNMS,SIZNMS,TYPLOC,PERNMS,
     +  INFLID,OTFLID
C
CXXXX INITIALIZE LOGICAL FLAGS
      ECHOFG=.FALSE.
      TABFG=.FALSE.
      FRQARV=.FALSE.
      WRTOUT=.FALSE.
      SETDIM=.FALSE.
      SETNMS=.FALSE.
C
CXXXX READ IN CONFIG FILE
      READ(NCRDR,FLAGNM)
      IF (SETDIM) READ(NCRDR,RDDIM)
      IF (SETNMS) READ(NCRDR,RSNMS)
      WRITE(NPRNT,FLAGNM)
      WRITE(NPRNT,RDDIM)
      CALL RDFACT(INFLID(10))
      IF (TABFG)  CALL PRTTAB
      RETURN
      END
CXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
C                                                                            X
C  Name: CONREQ                                                              X
C  Module Number: 1.6                                                        X
C  Function: Performs all subroutine calls to read input files to            X
C            determine weekly and periodic requirements and create           X
C            the constant ouput file.                                        X
C  Input Parameters: none                                                    X
C  Output Parameters: none                                                   X
C  Common Blocks/Variables Used: DIMEN/QTRNUM                                X
C      FILIDS/INFLID,OTFLID                                                  X
C      FLAGS/WRTOUT,FRQARV                                                   X
C      QTRDEF/QTRNMS                                                         X
C      UNITS/NINPUT                                                          X
C  Common Blocks/Variables Changed: none                                     X
C  Modules Called: WEEKLY,PERIOD,OPNINP,SRTWRT,CLSERR,ENDERR                 X
C  Calling Modules: MAIN                                                     X
C                                                                            X
CXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
C
      SUBROUTINE CONREQ
      CHARACTER FILENM*7
C
CXXXX COMMON BLOCKS
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +    NUMAD,NUMOTH,QTRNUM,WPCLAS
      CHARACTER*1 INFLID,OTFLID
      COMMON/FILIDS/ INFLID(10),OTFLID
      LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
```

```
      CHARACTER*2 QTRNMS
      COMMON/QTRDEF/ QTRNMS(5)
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
C**** READ AND PROCESS WEEKLY REQUIREMENTS FILE
      ICOUNT=0
      FILENM='WEKREQ'//INFLID(7)
      CALL OPNINP(FILENM,'F')
      CALL WEEKLY(FILENM,7,ICOUNT)
      CLOSE (NINPUT,ERR=200)
C
C**** READ AND PROCESS PERIODIC REQUIREMENTS FILE
      FILENM='PERREQ'//INFLID(8)
      CALL OPNINP(FILENM,'F')
      CALL PERIOD(FILENM,8,ICOUNT)
      CLOSE(NINPUT,ERR=200)
C
C**** OPEN AND WRITE CONSTANT FILE
      FILENM='CNST'//QTRNMS(QTRNUM)(:2)//OTFLID(:1)
      IF (WRTOUT) THEN
          CALL OPNOUT(FILENM,'U')
          CALL SRTWRT(ICOUNT,FILENM)
          CALL CLSFIL(FILENM)
      ELSE
          FILENM='NONE    '
          IF (FRQARV) CALL SRTWRT(ICOUNT,FILENM)
      ENDIF
      RETURN
C
  200 CALL CLSERR(FILENM)
      END
C*************************************************************************
C                                                                        *
C  Name: DAYSFT                                                          *
C  Module Number: 1.C.4                                                  *
C  Function: Calls PRTHDR to print format HDRNUM; opens format 3         *
C            FILENM; calls RDFMT3  to input file; and calls DSTWRK       *
C            to distribute the work over each week.                      *
C  Input Parameters: FILENM, HDRNUM                                      *
C  Output Parameters: none                                               *
C  Common Blocks/Variables Used: DIMEN/NUMWKS                            *
C      FLAGS/ECHOFG                                                      *
C      UNITS/NINPUT                                                      *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: RDFMT3,DSTWRK,PRTHDR,OPNINP,CLSERR                    *
C  Calling Modules: CALADM, CALOTH                                       *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE DAYSFT(FILENM,HDRNUM)
      CHARACTER FILENM*7
      INTEGER HDRNUM,CLASS,SIZE
      REAL INPC,INBA
C
C**** COMMON BLOCKS
```

```
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +     NUMAD,NUMOTH,QTRNUM,WPCLAS
      LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
      IF (ECHOFG)  CALL PRTHDR(FILENM,HDRNUM)
C
C**** OPEN AND PROCESS FORMAT 3 FILES
      INPC=0.0
      CALL OPNINP(FILENM,'F')
   10 CALL RDFMT3(LOCAT,INCOMP,CLASS,SIZE,NUMUSE,INBA)
      IF (LOCAT .EQ. 0) THEN
          CLOSE(NINPUT,ERR=300)
          RETURN
      ENDIF
C
C**** DISTRIBUTE DAILY ADMIN WORK
      DO 100  I=1,NUMWKS
          IWEEK=I
          CALL DSTWRK(LOCAT,IWEEK,NUMUSE,INCOMP,CLASS,SIZE,INPC,INBA)
  100 CONTINUE
      GO TO 10
C
  300 CALL CLSERR(FILENM)
      END
C*************************************************************************
C                                                                        *
C  Name: DISTWP                                                          *
C  Module Number: 1.C.5                                                  *
C  Function: Distributes the word processing load into TOTINT and        *
C            TOTPC for one week according to a word processing           *
C            factor for each computer and size load.                     *
C  Input Parameters: LOCAT,WEEK,XINT,XPC                                 *
C  Output Parameters: none                                               *
C  Common Blocks/Variables Used: DIMEN/NUMCPU,NUMSIZ                     *
C      FACTXX/WPFACT                                                     *
C      TOTALS/TOTINT,TOTPC                                               *
C  Common Blocks/Variables Changed: TOTALS/TOTINT,TOTPC                  *
C  Modules Called: none                                                  *
C  Calling Modules: WPWORK                                               *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE DISTWP(LOCAT,WEEK,XINT,XPC)
      PARAMETER (MAXLOC=3,MAXWKS=11,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      INTEGER WEEK
C
C**** COMMON BLOCKS
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +     NUMAD,NUMOTH,QTRNUM,WPCLAS
      COMMON/FACTXX/ PCFACT(MAXCLS,MAXSIZ),SESFACT(MAXCLS,MAXSIZ),
```

```
     +  RUNFACT(MAXCLS,MAXSIZ),BAFACT(MAXCPU),WPFACT(MAXCPU,MAXSIZ)
      COMMON/TOTALS/ TOTINT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +     TOTBAT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +     TOTPC(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ)
C
C**** DISTRIBUTE WORD PROCESSING LOAD
      DO 200  I=1,NUMCPU-1
          DO 100  J=1,NUMSIZ
              TOTINT(LOCAT,WEEK,I,WPCLAS,J)=TOTINT(LOCAT,
     +            WEEK,I,WPCLAS,J)+(XINT*WPFACT(I,J))
              TOTPC(LOCAT,WEEK,I,WPCLAS,J)=TOTPC(LOCAT,
     +            WEEK,I,WPCLAS,J)+(XPC*WPFACT(I,J))
  100     CONTINUE
  200 CONTINUE
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: DSTWRK                                                          *
C  Module Number: 1.C.6                                                  *
C  Function: Distributes a weekly work load into the TOTAL arrays        *
C            (TOTINT, TOTPC, TOTBAT) as follows:                         *
C                Batch users = (INBA*BAFACT(INCOMP))*INSTUD              *
C                PC users = INSTUD*INPC                                  *
C                Interactive users = INSTUD-(Batch users+PC users        *
C  Input Parameters: LOCAT,WEEKNO,INSTUD,INCOMP,CLASS,SIZE,INPC,INBA     *
C  Output Parameters: none                                               *
C  Common Blocks/Variables Used: FACTXX/BAFACT,PCFACT                    *
C      TOTALS/TOTINT,TOTBAT,TOTPC                                        *
C  Common Blocks/Variables Changed: TOTALS/TOTINT,TOTBAT,TOTPC           *
C  Modules Called: none                                                  *
C  Calling Modules: REGWRK, DAYSFT                                       *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE DSTWRK(LOCAT,WEEKNO,INSTUD,INCOMP,CLASS,SIZE,
     +     INPC,INBA)
      PARAMETER (MAXLOC=3,MAXWKS=11,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      REAL INPC,INBA,INTSTD
      INTEGER WEEKNO,CLASS,SIZE
C
C**** COMMON BLOCKS
      COMMON/FACTXX/ PCFACT(MAXCLS,MAXSIZ),SESFACT(MAXCLS,MAXSIZ),
     +  RUNFACT(MAXCLS,MAXSIZ),BAFACT(MAXCPU),WPFACT(MAXCPU,MAXSIZ)
      COMMON/TOTALS/ TOTINT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +     TOTBAT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +     TOTPC(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ)
C
C**** CALCULATE BATCH USERS
      TSTUD=REAL(INSTUD)
      IF ((INBA .GT. 0.0) .AND. (BAFACT(INCOMP) .GT. 0.0)) THEN
          TFACT=INBA*BAFACT(INCOMP)
          BATSTD=TSTUD*TFACT
          TOTBAT(LOCAT,WEEKNO,INCOMP,CLASS,SIZE)=
```

```
     +       TOTBAT(LOCAT,WEEKNO,INCOMP,CLASS,SIZE)+BATSTD
        ELSE
            BATSTD=0.0
        ENDIF
C
C**** CALCULATE PC USERS
        IF ((INPC .GT. 0.0) .AND. (PCFACT(CLASS,SIZE) .GT. 0.0)) THEN
            PCSTUD=INPC*PCFACT(CLASS,SIZE)*TSTUD
            TOTPC(LOCAT,WEEKNO,INCOMP,CLASS,SIZE)=
     +       TOTPC(LOCAT,WEEKNO,INCOMP,CLASS,SIZE)+PCSTUD
        ELSE
            PCSTUD=0.0
        ENDIF
C
C**** CALCULATE INTERACTIVE USERS
        INTSTD=TSTUD-(BATSTD+PCSTUD)
        IF (INTSTD .LE. 0.0) RETURN
        TOTINT(LOCAT,WEEKNO,INCOMP,CLASS,SIZE)=
     +     TOTINT(LOCAT,WEEKNO,INCOMP,CLASS,SIZE)+INTSTD
        RETURN
        END
C*************************************************************************
C                                                                        *
C  Name: FRQHDR                                                          *
C  Module Number: 1.C.7                                                  *
C  Function: Prints the arrival and frequency file headers for           *
C            OUTFIL.                                                     *
C  Input Parameters: OUTFIL,HDRNUM                                       *
C  Output Parameters: none                                               *
C  Common Blocks/Variables Used: UNITS/NPRNT                             *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: none                                                  *
C  Calling Modules: CALADM,CALSTD,BATCH ,CALOTH                          *
C                                                                        *
C*************************************************************************
C
        SUBROUTINE FRQHDR(OUTFIL,HDRNUM)
        CHARACTER*7 OUTFIL
        INTEGER HDRNUM
C
C**** COMMON BLOCKS
        COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
C**** PRINT HEADER FOR ARRIVAL AND FREQUENCY FILE
        GO TO (10,20,30,40) HDRNUM
        WRITE(NPRNT,5) HDRNUM
        STOP
     5 FORMAT('1','SUBROUTINE FRQHDR CALLED WITH UNKNOWN HEADER',
     +     ' NUMBER',I5)
C
    10 WRITE(NPRNT,15)
       WRITE(NPRNT,100) OUTFIL
       RETURN
    15 FORMAT('1','STUDENT ARRIVAL AND FREQUENCY FILE')
C
```

```
   20 WRITE(NPRNT,25)
      WRITE(NPRNT,100) OUTFIL
      RETURN
   25 FORMAT('1','FACULITY/ADMINSTRATION ARRIVAL AND FREQUENCY FILE')
C
   30 WRITE(NPRNT,35)
      WRITE(NPRNT,100) OUTFIL
      RETURN
   35 FORMAT('1','SOFTWARE DEVELOPMENT ARRIVAL AND FREQUENCY FILE')
C
   40 WRITE(NPRNT,45)
      WRITE(NPRNT,100) OUTFIL
      RETURN
   45 FORMAT('1','BATCH ARRIVAL AND FREQUENCY FILE')
C
  100 FORMAT(' ',' FILE NAME: ',A7)
      END
C*****************************************************************************
C                                                                            *
C  Name: OPNERR                                                              *
C  Module Number: 1.C.8                                                      *
C  Function: Prints filename when open error occurs and then stops           *
C            execution.                                                      *
C  Input Parameters: FILENM                                                  *
C  Output Parameters: none                                                   *
C  Common Blocks/Variables Used: UNITS/NPRNT                                 *
C  Common Blocks/Variables Changed: none                                     *
C  Modules Called: none                                                      *
C  Calling Modules: OPNINP,OPNOUT                                            *
C                                                                            *
C*****************************************************************************
C
      SUBROUTINE OPNERR(FILENM)
      CHARACTER FILENM*7
C
C**** COMMON BLOCKS
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
      WRITE(NPRNT,10) FILENM
      STOP
   10 FORMAT(' ','FILE ',A7,' OPEN ERROR')
      END
C*****************************************************************************
C                                                                            *
C  Name: OPNINP                                                              *
C  Module Number: 1.C.9                                                      *
C  Function: Opens the input files.                                          *
C  Input Parameters: INFIL,FORMAT                                            *
C  Output Parameters: none                                                   *
C  Common Blocks/Variables Used: UNITS/NINPUT                                *
C  Common Blocks/Variables Changed: none                                     *
C  Modules Called: OPNERR                                                    *
C  Calling Modules: CONREQ,REGWRK,WPWORK,DAYSFT                              *
C                                                                            *
C*****************************************************************************
```

```
C
      SUBROUTINE OPNINP(INFIL,FORMAT)
      CHARACTER INFIL*7,FORMAT*1
C
C**** COMMON BLOCKS
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
C**** OPEN AN INPUT FILE
      IF (FORMAT .EQ. 'U') THEN
        OPEN(UNIT=NINPUT,FILE=INFIL,STATUS='OLD',ACCESS='SEQUENTIAL',
     +       FORM='UNFORMATTED',ERR=100)
      ELSE
        OPEN(UNIT=NINPUT,FILE=INFIL,STATUS='OLD',ACCESS='SEQUENTIAL',
     +       FORM='FORMATTED',ERR=100)
      ENDIF
      RETURN
C
  100 CALL OPNERR(INFIL)
      END
C**********************************************************************
C                                                                     *
C  Name: OPNOUT                                                       *
C  Module Number: 1.C.10                                              *
C  Function: Opens the output files.                                  *
C  Input Parameters: OUTFIL,FORMAT                                    *
C  Output Parameters: none                                            *
C  Common Blocks/Variables Used: UNITS/NOUT                           *
C  Common Blocks/Variables Changed: none                              *
C  Modules Called: OPNERR                                             *
C  Calling Modules: BATCH,CALADM,CALOTH,CALSTD,CONREQ                 *
C                                                                     *
C**********************************************************************
C
      SUBROUTINE OPNOUT(OUTFIL,FORMAT)
      CHARACTER OUTFIL*7,FORMAT*1
C
C**** COMMON BLOCKS
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
C**** OPEN AND OUTPUT FILE
      IF (FORMAT .EQ. 'U') THEN
          OPEN(UNIT=NOUT,FILE=OUTFIL,STATUS='NEW',ACCESS='SEQUENTIAL',
     +       FORM='UNFORMATTED',ERR=100)
      ELSE
          OPEN(UNIT=NOUT,FILE=OUTFIL,STATUS='NEW',ACCESS='SEQUENTIAL',
     +       FORM='FORMATTED',ERR=100)
      ENDIF
      RETURN
C
  100 CALL OPNERR(OUTFIL)
      END
C**********************************************************************
C                                                                     *
C  Name: PERIOD                                                       *
C  Module Number: 1.6.2                                               *
```

```
C  Function: Calls PRTHDR to print header HDRNUM; opens file FILENM;*
C            calls RDFMT4 to read periodic requirements              *
C  Input Parameters: FILENM,HDRNUM,ICOUNT                            *
C  Output Parameters: none                                           *
C  Common Blocks/Variables Used: FLAGS/ECHOFG                        *
C      TOTALS/TOTINT,TOTBAT,TOTPC                                    *
C  Common Blocks/Variables Changed: Uses TOTALS space.               *
C  Modules Called: RDFMT4,PRTHDR                                     *
C  Calling Modules: CONREQ                                           *
C                                                                    *
C*******************************************************************
C
      SUBROUTINE PERIOD(FILENM,HDRNUM,ICOUNT)
      PARAMETER (MAXLOC=3,MAXWKS=11,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      CHARACTER FILENM*7
      DIMENSION SORTAR(1000,5),ISRTAR(1000,5)
      INTEGER  HDRNUM,CLASS,SIZE,DAYNO
C
C**** COMMON BLOCKS
      LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/TOTALS/ TOTINT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTBAT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTPC(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ)
      EQUIVALENCE (SORTAR(1,1),ISRTAR(1,1),TOTINT(1,1,1,1,1))
C
      IF (ECHOFG)  CALL PRTHDR(FILENM,HDRNUM)
C
C**** PROCESS FORMAT 4 (PERIODIC) FILE
   10 CALL RDFMT4(LOCNUM,TIMDAY,INCOMP,CLASS,SIZE,DAYNO)
      IF (LOCNUM .EQ. 0)  RETURN
      XTIME=(REAL(DAYNO)*86400.)+TIMDAY*3600.
      ICOUNT=ICOUNT+1
      SORTAR(ICOUNT,1)=XTIME
      ISRTAR(ICOUNT,2)=LOCNUM
      ISRTAR(ICOUNT,3)=INCOMP
      ISRTAR(ICOUNT,4)=CLASS
      ISRTAR(ICOUNT,5)=SIZE
      GO TO 10
C
      END
C*******************************************************************
C                                                                    *
C  Name: PRTHDR                                                      *
C  Module Number: 1.C.11                                             *
C  Function: Prints the echo input headers.                          *
C  Input Parameters: FILENM,NUMHDR                                   *
C  Output Parameters: none                                           *
C  Common Blocks/Variables Used: DIMEN/QTRNUM                        *
C      QTRDEF/QTRNMS                                                 *
C      UNITS/NPRNT                                                   *
C  Common Blocks/Variables Changed: none                             *
C  Modules Called: none                                              *
C  Calling Modules: WEEKLY,PERIOD,REGWRK,WPWORK,DAYSFT               *
```

```
C                                                                       *
C***********************************************************************
C
      SUBROUTINE PRTHDR(FILENM,NUMHDR)
C
      CHARACTER FILENM*7
      INTEGER NUMHDR
C
C**** COMMON BLOCKS
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +    NUMAD,NUMOTH,QTRNUM,WPCLAS
      CHARACTER*2 QTRNMS
      COMMON/QTRDEF/ QTRNMS(5)
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
C**** PRINT ECHO HEADERS
      GO TO (10,20,30,40,50,60,70,80,90) NUMHDR
      WRITE(NPRNT,5)
      RETURN
    5 FORMAT('1','*** ERROR *** SUBROUTINE PRTHDR CALLED WITH BAD ',
     +    'UNKNOWN HEADER NUMBER')
C
C**** PRINT STUDENT COURSE WORK
   10 WRITE(NPRNT,15)  FILENM,QTRNMS(QTRNUM)
      RETURN
   15 FORMAT('1',56X,'STUDENT COURSE WORK',/,
     +   49X,'FILE NAME:',A7,2X,A6,1X,'QUARTER',//,
     +   30X,'COURSE  NUM  OF',15X,'REQUIREMENT/WEEK',19X,'PC',
     +   5X,'BATCH',/,
     +   23X,'LOCAT  NUMBER  STUDENTS  1   2   3   4   5   6   7   8',
     +   '   9  10  11    FACTOR   FACTOR')
C
C**** PRINT STUDENT WORD PROCESSING LOAD
   20 WRITE(NPRNT,25)  FILENM,QTRNMS(QTRNUM)
      RETURN
   25 FORMAT('1',53X,'STUDENT WORD PROCESSING LOAD',/,
     +   51X,'FILE NAME:',A7,2X,A6,1X,'QUARTER',//,
     +   70X,'REQUIREMENTS/WEEK',27X,'PC',/,
     +   29X,'LOCAT    STUDENTS PERCNT    1    2    3    4    5',
     +   '    6    7    8    9   10   11    FACTOR')
C
C**** PRINT STUDENT THESIS REQUIREMENTS
   30 WRITE(NPRNT,35)  FILENM,QTRNMS(QTRNUM)
      RETURN
   35 FORMAT('1',53X,'STUDENT THESIS REQUIREMENTS',/,
     +   51X,'FILE NAME:',A7,2X,A6,1X,'QUARTER',//,
     +   39X,'NUM OF',15X,'REQUIREMENT/WEEK',19X,'PC',
     +   4X,'BATCH',/,
     +   22X,'LOCAT   DESCRP  STUDENTS  1   2   3   4   5   6   7   8',
     +   '   9  10  11    FACTOR  FACTOR')
C
C**** PRINT FACULITY RESEARCH REQUIREMENTS
   40 WRITE(NPRNT,45)  FILENM,QTRNMS(QTRNUM)
      RETURN
```

```
   45 FORMAT('1',51X,'FACULITY RESEARCH REQUIREMENTS',/,
     +   51X,'FILE NAME:',A7,2X,A6,1X,'QUARTER',//,
     +   39X,'NUM OF',15X,'REQUIREMENT/WEEK',19X,'PC',
     +   5X,'BATCH',/,
     +   22X,'LOCAT   DESCRP  FACULITY  1   2   3   4   5   6   7   8',
     +   '   9  10  11    FACTOR  FACTOR')
C
C**** PRINT SOFTWARE DEVELOPMENT REQUIREMENTS
   50 WRITE(NPRNT,55)  FILENM,QTRNMS(QTRNUM)
      RETURN
   55 FORMAT('1',50X,'SOFTWARE DEVELOPMENT REQUIREMENTS',/,
     +   51X,'FILE NAME:',A7,2X,A6,1X,'QUARTER',//,
     +   85X,'# DEV   BATCH',/,
     +   34X, 'LOCATION    PROGID  COMPUTERS   CLASS      SIZE',
     +   '     TEAM  FACTOR')
C
C**** PRINT DAILY ADMINSTRATIVE REQUIREMENTS
   60 WRITE(NPRNT,65)  FILENM,QTRNMS(QTRNUM)
      RETURN
   65 FORMAT('1',50X,'DAILY ADMINSTRATIVE REQUIREMENTS',/,
     +   51X,'FILE NAME:',A7,2X,A6,1X,'QUARTER',//,
     +   85X,'# OF    BATCH',/,
     +   36X, 'LOCAT    DESCRP   COMPUTERS  CLASS       SIZE    USERS',
     +   '  FACTOR')
C
C**** PRINT WEEKLY ADMINSTRATIVE REQUIREMENTS
   70 WRITE(NPRNT,75)  FILENM,QTRNMS(QTRNUM)
      RETURN
   75 FORMA'('1',50X,'WEEKLY ADMINSTRATIVE REQUIREMENTS',/,
     +   51X,'FILE NAME:',A7,2X,A6,1X,'QUARTER',//,
     +   26X, 'LOCATION   DESCRP   COMPUTER     CLASS     SIZE       ',
     +   'DAY#    TIMDAY')
C
C**** PRINT PERIODIC ADMINSTRATIVE REQUIREMENTS
   80 WRITE(NPRNT,85)  FILENM,QTRNMS(QTRNUM)
      RETURN
   85 FORMAT('1',49X,'PERIODIC ADMINSTRATIVE REQUIREMENTS',/,
     +   51X,'FILE NAME:',A7,2X,A6,1X,'QUARTER',//,
     +   26X, 'LOCATION   DESCRP   COMPUTER     CLASS     SIZE       ',
     +   'DAY#    TIMDAY')
C
C**** PRINT FACULITY WORD PROCESSING REQUIREMENTS
   90 WRITE(NPRNT,95)  FILENM,QTRNMS(QTRNUM)
      RETURN
   95 FORMAT('1',53X,'FACULITY WORD PROCESSING LOAD',/,
     +   51X,'FILE NAME:',A7,2X,A6,1X,'QUARTER',//,
     +   70X,'REQUIREMENTS/WEEK',27X,'PC',/,
     +   29X,'LOCAT    FACULITY PERCNT    1    2    3    4    5',
     +   '    6    7    8    9   10   11    FACTOR')
      END
C*************************************************************************
C                                                                        *
C  Name: PRTTAB                                                          *
C  Module Number: 1.1.2                                                  *
C  Function: Prints the input FACTOR file tables and computer,           *
```

```
C            class, and size descriptions.                              *
C  Input Parameters: none                                               *
C  Output Parameters: none                                              *
C  Common Blocks/Variables Used: DAYSXX/DAYSST,DAYSAD,DAYSBA            *
C       DIMEN/NUMCPU,NUMCLS,NUMSIZ                                      *
C       FACTXX/SESFACT,PCFACT,RUNFACT,BAFACT,WPFACT                     *
C       HOURXX/HOURST,HOURAD,HOURBA                                     *
C       IDS/COMPID,CLASID,SIZIDS                                        *
C       NAMES/COMPNM,CLASNM,SIZNMS,TYPLOC                               *
C       UNITS/NPRNT                                                     *
C  Common Blocks/Variables Changed:none                                 *
C  Modules Called: none                                                 *
C  Calling Modules: CONFIG                                              *
C                                                                       *
C***********************************************************************
C
      SUBROUTINE PRTTAB
      PARAMETER (MAXLOC=3,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
C**** COMMON BLOCKS
      COMMON/DAYSXX/ DAYSST(7),DAYSAD(7),DAYSBA(7)
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +    NUMAD,NUMOTH,QTRNUM,WPCLAS
      COMMON/FACTXX/ PCFACT(MAXCPU,MAXSIZ),SESFACT(MAXCPU,MAXSIZ),
     +  RUNFACT(MAXCPU,MAXSIZ),BAFACT(MAXCPU),WPFACT(MAXCPU,MAXSIZ)
      COMMON/HOURXX/  HOURST(24),HOURAD(24),HOURBA(24)
      CHARACTER*1 COMPID,CLASID,SIZIDS
      COMMON/IDS/ COMPID(MAXCPU),CLASID(MAXCLS),SIZIDS(MAXSIZ)
      CHARACTER*8 CLASNM,COMPNM,LOCNMS,SIZNMS,TYPLOC,PERNMS
      COMMON/NAMES/ CLASNM(MAXCLS),COMPNM(MAXCPU),LOCNMS(MAXLOC),
     +    SIZNMS(MAXSIZ),TYPLOC(MAXLOC),PERNMS(10)
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
      WRITE(NPRNT,110)
C
C**** PRINT DAILY FACTOR TABLE
      WRITE(NPRNT,120)
      WRITE(NPRNT,130)  TYPLOC(1),DAYSST
      WRITE(NPRNT,130)  TYPLOC(2),DAYSBA
      WRITE(NPRNT,130)  TYPLOC(3),DAYSAD
C
C**** PRINT HOURLY FACTOR TABLES
      WRITE(NPRNT,140)
      WRITE(NPRNT,150)  TYPLOC(1),HOURST
      WRITE(NPRNT,150)  TYPLOC(2),HOURBA
      WRITE(NPRNT,150)  TYPLOC(3),HOURAD
C
C**** IDENTIFY COMPUTERS AND CODES
      WRITE(NPRNT,160)
      DO 10  I=1,NUMCPU
          WRITE(NPRNT,170)  I,COMPID(I),COMPNM(I)
   10 CONTINUE
C
C**** IDENTIFY CLASSES AND CODES
```

```
      WRITE(NPRNT,180)
      DO 20  I=1,NUMCLS
          WRITE(NPRNT,170)  I,CLASID(I),CLASNM(I)
   20 CONTINUE
C
CXXXX IDENTIFY MEMORY SIZES AND CODES
      WRITE(NPRNT,190)
      DO 30  I=1,NUMSIZ
          WRITE(NPRNT,170)  I,SIZIDS(I),SIZNMS(I)
   30 CONTINUE
C
CXXXX PRINT FACTOR TABLE
      WRITE(NPRNT,210)
      DO 50  I=1,NUMCLS
          WRITE(NPRNT,220)  CLASNM(I)
          DO 40  J=1,NUMSIZ
              WRITE(NPRNT,230)  SIZNMS(J),SESFACT(I,J),PCFACT(I,J),
     +          RUNFACT(I,J)
   40     CONTINUE
   50 CONTINUE
C
CXXXX PRINT WORD PROCESSING FACTORS
      WRITE(NPRNT,250)
      DO 70  I=1,NUMCPU-1
          WRITE(NPRNT,260)  COMPNM(I)
          DO 60  J=1,NUMSIZ
              WRITE(NPRNT,270)  SIZNMS(J),WPFACT(I,J)
   60     CONTINUE
   70 CONTINUE
C
CXXXX PRINT BATCH FACTOR
      WRITE(NPRNT,280)
      DO 80  I=1,NUMCPU-1
          WRITE(NPRNT,290) COMPNM(I),BAFACT(I)
   80 CONTINUE
      RETURN
C
CXXXX FORMAT STATEMENTS
  110 FORMAT('1',59X,'INPUT TABLES')
  120 FORMAT('0',55X,'DISTRIBUTION OVER WEEK',/,
     +    38X,'LOCATION   MON    TUE    WED    THR    FRI    SAT    ',
     +    'SUN')
  130 FORMAT(' ',37X,A8,7F7.3)
  140 FORMAT('0',55X,'DISTRIBUTION OVER DAY',/,
     +    1X,'LOCATION     0000   0100   0200   0300   0400   0500   ',
     +  '0600   0700   0800   0900   1000   1100',/,12X,'   1200   ',
     +  '1300   1400   1500   1600   1700   1800   1900   2000   2100',
     +  '   2200   2300')
  150 FORMAT(' ',A8,3X,12F7.3,/,12X,12F7.3)
  160 FORMAT('0',54X,'COMPUTERS DESCRIPTION',/,
     +    54X,'NUMBER  CODE  DESCRIPTION')
  170 FORMAT(' ',54X,I4,6X,A1,5X,A8)
  180 FORMAT('0',54X,'CLASS WORK DESCRIPTION',/,
     +    54X,'NUMBER  CODE  DESCRIPTION')
  190 FORMAT('0',54X,'MEMORY SIZE DESCRIPTIONS',/,
```

```
     +    54X,'NUMBER  CODE  DESCRIPTION')
  210 FORMAT('1',60X,'FACTOR TABLE',/,
     +    57X,'SESSION     PC       # OF',/,
     +    36X,'CLASS      SIZE      FACTOR     FACTOR   BATCH')
  220 FORMAT(' ',34X,A8)
  230 FORMAT('+',44X,A8,3F10.4,/)
  250 FORMAT('1',54X,'WORD PROCESSING FACTORS',/,
     +    53X,'DISTRIBUTION OVER COMPUTERS',/,
     +    52X,'COMPUTER     SIZE      FACTOR')
  260 FORMAT(' ',52X,A8)
  270 FORMAT('+',62X,A8,F10.4,/)
  280 FORMAT('0',60X,'BATCH FACTOR',/,58X,'COMPUTER  FACTOR')
  290 FORMAT(' ',58X,A8,F6.1)
      END
C*************************************************************************
C                                                                        *
C  Name: RDERR                                                           *
C  Module Number: 1.C.12                                                 *
C  Function: Prints the unit number when a read error occurs.            *
C  Input Parameters: IUNIT                                               *
C  Output Parameters: none                                               *
C  Common Blocks/Variables Used: UNITS/NPRNT                             *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: none                                                  *
C  Calling Modules: RDFMT1-4                                             *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE RDERR(IUNIT)
C
CXXXX COMMON BLOCKS
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
      WRITE(NPRNT,10)  IUNIT
      STOP
   10 FORMAT(' ','UNIT ',I2,' READ ERROR')
      END
C*************************************************************************
C                                                                        *
C  Name: RDFACT                                                          *
C  Module Number: 1.1.1                                                  *
C  Function: Opens and reads FACTORx file.                               *
C  Input Parameters: FILEID                                              *
C  Output Parameters: none                                               *
C  Common Blocks/Variables Used: DAYSXX/DAYSST,DAYSAD,DAYSBA             *
C       DIMEN/NUMCPU,NUMCLS,NUMSIZ                                       *
C       FACTXX/PCFACT,BAFACT,WPFACT,SESFACT,RUNFACT                      *
C       HOURXX/HOURST,HOURAD,HOURBA                                      *
C       UNITS/NINPUT                                                     *
C  Common Blocks/Variables Changed: DAYSXX/DAYSST,DAYSAD,DAYSBA          *
C       FACTXX/PCFACT,BAFACT,WPFACT,SESFACT,RUNFACT                      *
C       HOURXX/HOURST,HOURAD,HOURBA                                      *
C  Modules Called:   none                                                *
C  Calling Modules: CONFIG                                               *
C                                                                        *
```

```
C**********************************************************************
C
      SUBROUTINE RDFACT(FILEID)
      PARAMETER (MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      CHARACTER FILEID*1,FILENM*7
C
C**** COMMON BLOCKS
      COMMON/DAYSXX/ DAYSST(7),DAYSAD(7),DAYSBA(7)
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +    NUMAD,NUMOTH,QTRNUM,WPCLAS
      COMMON/FACTXX/ PCFACT(MAXCPU,MAXSIZ),SESFACT(MAXCPU,MAXSIZ),
     +  RUNFACT(MAXCPU,MAXSIZ),BAFACT(MAXCPU),WPFACT(MAXCPU,MAXSIZ)
      COMMON/HOURXX/  HOURST(24),HOURAD(24),HOURBA(24)
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
C**** OPEN FACTOR FILE
      FILENM='FACTOR'//FILEID(:1)
      CALL OPNINP(FILENM,'F')
C
C**** READ FACTOR FILE
      READ(NINPUT,10) I,J,K
      IF ((I.NE.NUMCPU).OR.(J.NE.NUMCLS).OR.(K.NE.NUMSIZ)) THEN
          WRITE(NPRNT,5)
          STOP
      ENDIF
      READ(NINPUT,*)
      READ(NINPUT,20)  DAYSST,DAYSAD,DAYSBA
      READ(NINPUT,*)
      READ(NINPUT,30)  HOURST,HOURAD,HOURBA
      READ(NINPUT,*)
      DO 200  I=1,NUMCLS
          DO 100  J=1,NUMSIZ
              READ(NINPUT,20)  SESFACT(I,J),PCFACT(I,J),RUNFACT(I,J)
  100     CONTINUE
  200 CONTINUE
      READ(NINPUT,*)
      DO 300  I=1,NUMCPU
          READ(NINPUT,20)  BAFACT(I)
  300 CONTINUE
      READ(NINPUT,*)
      DO 500  I=1,NUMCPU-1
          READ(NINPUT,20)  (WPFACT(I,J), J=1,NUMSIZ)
  500 CONTINUE
      CLOSE(NINPUT,ERR=700)
      RETURN
  700 CALL CLSERR(FILENM)
C
C**** FORMAT STATEMENTS
    5 FORMAT('1','FACTOR TABLES DOES NOT MATCH INPUT DIMENSIONS')
   10 FORMAT(3I5)
   20 FORMAT(10X,7F10.4)
   30 FORMAT(10X,8F8.4)
      END
```

```
C*************************************************************************
C                                                                        *
C  Name: RDFMT1                                                          *
C  Module Number: 1.C.13                                                 *
C  Function: Reads and edits format 1 input files.                       *
C  Input Parameters: none                                                *
C  Output Parameters: LOCAT,INSTUD,WORK,INPC,INBA                        *
C  Common Blocks/Variables Used: DIMEN/QTRNUM                            *
C      FLAGS/ECHOFG                                                      *
C      NAMES/LOCNMS                                                      *
C      UNITS/NPRNT,NINPUT                                                *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: RDERR                                                 *
C  Calling Modules: REGWRK                                               *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE RDFMT1(LOCAT,INSTUD,WORK,INPC,INBA)
      PARAMETER (MAXLOC=3,MAXWKS=11,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      CHARACTER QTR(4)*1,COURSE*7,WORK(3,MAXWKS)*1
      REAL INPC,INBA
      INTEGER  TSTUDS(4)
C
C**** COMMON BLOCKS
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +     NUMAD,NUMOTH,QTRNUM,WPCLAS
      LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
      CHARACTER*8 CLASNM,COMPNM,LOCNMS,SIZNMS,TYPLOC,PERNMS
      COMMON/NAMES/ CLASNM(MAXCLS),COMPNM(MAXCPU),LOCNMS(MAXLOC),
     +     SIZNMS(MAXSIZ),TYPLOC(MAXLOC),PERNMS(10)
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
   10 READ(NINPUT,500,END=100,ERR=200)  LOCAT,COURSE,QTR,TSTUDS,WORK,
     +     INPC,INBA
      IF (QTR(QTRNUM) .EQ. 'X') THEN
          INSTUD=TSTUDS(QTRNUM)
          IF (ECHOFG) WRITE(NPRNT,510) LOCNMS(LOCAT),COURSE,
     +         INSTUD,WORK,INPC,INBA
          RETURN
      ENDIF
      GO TO 10
C
  100 LOCAT=0
      RETURN
  200 CALL RDERR(NINPUT)
C
C**** FORMAT STATEMENTS
  500 FORMAT(I1,1X,A7,1X,4A1,1X,4I2,11(1X,A1,A1,A1),2(1X,F4.2))
  510 FORMAT(' ',20X,A8,1X,A7,I6,3X,11(1X,A1,A1,A1),2F8.3)
      END
C*************************************************************************
C                                                                        *
```

```
C  Name: RDFMT2                                                          *
C  Module Number: 1.C.14                                                 *
C  Function: Reads and edits format 2 input files.                       *
C  Input Parameters: none                                                *
C  Output Parameters: LOCAT,INSTUD,PERCNT,LDFACT,INPC                    *
C  Common Blocks/Variables Used: DIMEN/QTRNUM                            *
C       FLAGS/ECHOFG                                                     *
C       NAMES/LOCNMS                                                     *
C       QTRDEF/QTRNMS                                                    *
C       UNITS/NPRNT,NINPUT                                               *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: RDERR                                                 *
C  Calling Modules: WPWORK                                               *
C                                                                        *
C************************************************************************
C
      SUBROUTINE RDFMT2(LOCAT,INSTUD,PERCNT,LDFACT,INPC)
      PARAMETER (MAXLOC=3,MAXWKS=11,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      REAL INPC,PERCNT,LDFACT(MAXWKS)
      CHARACTER QTR*2
C
C**** COMMON BLOCKS
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +     NUMAD,NUMOTH,QTRNUM,WPCLAS
      LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
      CHARACTER*8 CLASNM,COMPNM,LOCNMS,SIZNMS,TYPLOC,PERNMS
      COMMON/NAMES/ CLASNM(MAXCLS),COMPNM(MAXCPU),LOCNMS(MAXLOC),
     +     SIZNMS(MAXSIZ),TYPLOC(MAXLOC),PERNMS(10)
      CHARACTER*2 QTRNMS
      COMMON/QTRDEF/ QTRNMS(5)
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
   10 READ(NINPUT,500,END=100,ERR=200)  LOCAT,QTR,INSTUD,PERCNT,
     +     LDFACT,INPC
      IF (QTR .EQ. QTRNMS(QTRNUM)) THEN
          IF (ECHOFG) WRITE(NPRNT,510) LOCNMS(LOCAT),INSTUD,
     +          PERCNT,LDFACT,INPC
          RETURN
      ENDIF
      GO TO 10
C
  100 LOCAT=0
      RETURN
  200 CALL RDERR(NINPUT)
C
C**** FORMAT STATEMENTS
  500 FORMAT(I1,1X,A2,I5,F6.3,11F4.2,F5.2)
  510 FORMAT(' ',26X,A8,I8,F9.2,2X,11(F5.2),F8.3)
      END
C************************************************************************
C                                                                        *
C  Name: RDFMT3                                                          *
```

```
C  Module Number: 1.C.15                                                  *
C  Function: Reads and edits format 3 input files.                        *
C  Input Parameters: none                                                  *
C  Output Parameters: LOCAT,INCOMP,CLASS,SIZE,NUMUSE,INBA                 *
C  Common Blocks/Variables Used: DIMEN/QTRNUM                             *
C       FLAGS/ECHOFG                                                       *
C       NAMES/LOCNMS,COMPNM,CLASNM,SIZNMS                                  *
C       QTRDEF/QTRNMS                                                      *
C       UNITS/NPRNT,NINPUT                                                 *
C  Common Blocks/Variables Changed: none                                   *
C  Modules Called: FIGCPU,FIGCLS,FIGSIZ,RDERR                              *
C  Calling Modules: DAYSFT                                                 *
C                                                                          *
C**************************************************************************
C
      SUBROUTINE RDFMT3(LOCAT,INCOMP,CLASS,SIZE,NUMUSE,INBA)
      PARAMETER (MAXLOC=3,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      CHARACTER PROGID*10,QTR*2,AMACH*1,ACLASS*1,ASIZE*1
      REAL INBA
      INTEGER  FIGCLS,FIGCPU,FIGSIZ
      INTEGER  CLASS,SIZE
C
C**** COMMON BLOCKS
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +    NUMAD,NUMOTH,QTRNUM,WPCLAS
      LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
      CHARACTER*8 CLASNM,COMPNM,LOCNMS,SIZNMS,TYPLOC,PERNMS
      COMMON/NAMES/ CLASNM(MAXCLS),COMPNM(MAXCPU),LOCNMS(MAXLOC),
     +    SIZNMS(MAXSIZ),TYPLOC(MAXLOC),PERNMS(10)
      CHARACTER*2 QTRNMS
      COMMON/QTRDEF/ QTRNMS(5)
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
   10 READ(NINPUT,500,END=100,ERR=200)  LOCAT,PROGID,QTR,AMACH,ACLASS,
     +    ASIZE,NUMUSE,INBA
      IF ((QTR.EQ.QTRNMS(5)).OR.(QTR.EQ.QTRNMS(QTRNUM))) THEN
          INCOMP=FIGCPU(AMACH)
          CLASS=FIGCLS(ACLASS)
          SIZE=FIGSIZ(ASIZE)
          IF (ECHOFG) WRITE(NPRNT,510) LOCNMS(LOCAT),PROGID,
     +        COMPNM(INCOMP),CLASNM(CLASS),SIZNMS(SIZE),NUMUSE,INBA
          RETURN
      ENDIF
      GO TO 10
C
  100 LOCAT=0
      RETURN
  200 CALL RDERR(NINPUT)
C
C**** FORMAT STATEMENTS
  500 FORMAT(I1,1X,A10,1X,A2,1X,3A1,1X,I3,F6.3)
  510 FORMAT(' ',34X,5(A8,2X),I4,F9.3)
```

```
      END
C*************************************************************************
C                                                                        *
C  Name: RDFMT4                                                          *
C  Module Number: 1.C.16                                                 *
C  Function: Reads and edits format 4 input files.                       *
C  Input Parameters: none                                                *
C  Output Parameters: LOCNUM,TIMDAY,INCOMP,CLASS,SIZE,DAYNO              *
C  Common Blocks/Variables Used: DIMEN/QTRNUM                            *
C       FLAGS/ECHOFG                                                     *
C       NAMES/COMPNM,CLASNM,SIZNMS,PERNMS                                *
C       QTRDEF/QTRNMS                                                    *
C       UNITS/NPRNT,NINPUT                                               *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: FIGCPU,FIGCLS,FIGSIZ,RDERR                            *
C  Calling Modules: WEEKLY,PERIOD                                        *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE RDFMT4(LOCNUM,TIMDAY,INCOMP,CLASS,SIZE,DAYNO)
      PARAMETER (MAXLOC=3,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      CHARACTER DESCRP*10,QTR*2,AMACH*1,ACLASS*1,ASIZE*1
      INTEGER  FIGCLS,FIGCPU,FIGSIZ
      INTEGER  CLASS,SIZE,DAYNO
C
C**** COMMON BLOCKS
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +    NUMAD,NUMOTH,QTRNUM,WPCLAS
      LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
      CHARACTER*8 CLASNM,COMPNM,LOCNMS,SIZNMS,TYPLOC,PERNMS
      COMMON/NAMES/ CLASNM(MAXCLS),COMPNM(MAXCPU),LOCNMS(MAXLOC),
     +    SIZNMS(MAXSIZ),TYPLOC(MAXLOC),PERNMS(10)
      CHARACTER*2 QTRNMS
      COMMON/QTRDEF/ QTRNMS(5)
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
   10 READ(NINPUT,500,END=100,ERR=200)  LOCNUM,DESCRP,QTR,AMACH,ACLASS,
     +    ASIZE,DAYNO,TIMDAY
      IF ((QTR.EQ.QTRNMS(5)).OR.(QTR.EQ.QTRNMS(QTRNUM))) THEN
          INCOMP=FIGCPU(AMACH)
          CLASS=FIGCLS(ACLASS)
          SIZE=FIGSIZ(ASIZE)
          IF (ECHOFG) WRITE(NPRNT,510) PERNMS(LOCNUM),DESCRP,
     +      COMPNM(INCOMP),CLASNM(CLASS),SIZNMS(SIZE),DAYNO,TIMDAY
          RETURN
      ENDIF
      GO TO 10
C
  100 LOCNUM=0
      RETURN
  200 CALL RDERR(NINPUT)
C
```

```
C**** FORMAT STATEMENTS
  500 FORMAT(I2,1X,A10,1X,A2,1X,3A1,I5,F10.4)
  510 FORMAT(' ',25X,A8,3X,4(A8,2X),I6,F12.4)
      END
C*************************************************************************
C                                                                        *
C  Name: REGWRK                                                          *
C  Module Number: 1.C.17                                                 *
C  Function: Calls PRTHDR to print header HDRNUM; calls RDFMT1 to        *
C            read format 1 input files and distribute the input          *
C            workload over the weeks by calling DSTWRK.                  *
C  Input Parameters: FILENM,HDRNUM                                       *
C  Output Parameters: none                                               *
C  Common Blocks/Variables Used: DIMEN/NUMWKS                            *
C      FLAGS/ECHOFG                                                      *
C      UNITS/NINPUT                                                      *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: DSTWRK,RDFMT1,PRTHDR,FIGCPU,FIGCLS,FIGSIZ,CLSERR      *
C  Calling Modules:  CALSTD,CALADM                                       *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE REGWRK(FILENM,HDRNUM)
      PARAMETER (MAXWKS=11)
C
      CHARACTER FILENM*7,WORK(3,MAXWKS)*1
      REAL  INPC,INBA
      INTEGER  FIGCLS,FIGCPU,FIGSIZ
      INTEGER  HDRNUM,CLASS,SIZE
C
C**** COMMON BLOCKS
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +    NUMAD,NUMOTH,QTRNUM,WPCLAS
      LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
      CALL OPNINP(FILENM,'F')
      IF (ECHOFG) CALL PRTHDR(FILENM,HDRNUM)
C
   10 CALL RDFMT1(LOCAT,INSTUD,WORK,INPC,INBA)
      IF (LOCAT .EQ. 0) THEN
          CLOSE(NINPUT,ERR=300)
          RETURN
      ENDIF
      DO 100  I=1,NUMWKS
          IF (WORK(1,I) .EQ. ' ') GO TO 100
          INCOMP=FIGCPU(WORK(1,I))
          CLASS=FIGCLS(WORK(2,I))
          SIZE=FIGSIZ(WORK(3,I))
          IWEEK=I
          CALL DSTWRK(LOCAT,IWEEK,INSTUD,INCOMP,CLASS,SIZE,INPC,INBA)
  100 CONTINUE
      GO TO 10
```

```
C
  300 CALL CLSERR(FILENM)
      END
C*****************************************************************************
C                                                                             *
C  Name: SRTWRT                                                               *
C  Module Number: 1.6.3                                                       *
C  Function: Sorts weekly and periodic records on time, then writes           *
C            output file CONSTy.                                              *
C  Input Parameters: ICOUNT,FILENM                                            *
C  Output Parameters: none                                                    *
C  Common Blocks/Variables Used: FLAGS/WRTOUT,FRQARV                          *
C     UNITS/NPRNT,NOUT                                                        *
C     TOTALS/space used                                                       *
C  Common Blocks/Variables Changed: none                                      *
C  Modules Called: none                                                       *
C  Calling Modules: CONREQ                                                    *
C                                                                             *
C*****************************************************************************
C
      SUBROUTINE SRTWRT(ICOUNT,FILENM)
      PARAMETER (MAXLOC=3,MAXWKS=11,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      CHARACTER FILENM*7
      DIMENSION SORTAR(1000,5),ISRTAR(1000,5),ISORT(1000)
C
C**** COMMON BLOCKS
      LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/TOTALS/ TOTINT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTBAT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTPC(MAXLOC.MAXWKS,MAXCPU,MAXCLS,MAXSIZ)
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
      EQUIVALENCE (SORTAR(1,1),ISRTAR(1,1),TOTINT(1,1,1,1,1))
C
      SWITCH=1.
      DO 100  I=1,ICOUNT
          ISORT(I)=I
  100 CONTINUE
  200 IF (SWITCH .EQ. 1.) THEN
          SWITCH=0.
          DO 300  I=2,ICOUNT
              IF (SORTAR(ISORT(I),1) .LT. SORTAR(ISORT(I-1),1)) THEN
                  ITEMP=ISORT(I)
                  ISORT(I)=ISORT(I-1)
                  ISORT(I-1)=ITEMP
                  SWITCH=1.
              ENDIF
  300     CONTINUE
          GO TO 200
      ENDIF
C
C**** WRITE CONSTANT FILE
      IF (WRTOUT) THEN
          DO 400   I=1,ICOUNT
```

```
            WRITE(NOUT) SORTAR(ISORT(I),1),(ISRTAR(ISORT(I),J),J=2,5)
  400     CONTINUE
      ENDIF
      IF (FRQARV) THEN
          WRITE(NPRNT,500) FILENM
          DO 450  I=1,ICOUNT
              WRITE(NPRNT,510)  SORTAR(ISORT(I),1),
     +          (ISRTAR(ISORT(I),J), J=2,5)
  450     CONTINUE
      ENDIF
      RETURN
C
C**** FORMAT STATEMENTS
  500 FORMAT('1','CONSTANT FILE: ',A7,/,37X,'TIME',10X,
     +    'NUMBER   COMPUTER   CLASS   SIZE')
  510 FORMAT(' ',32X,F10.1,4I10)
      END
C*************************************************************************
C                                                                        *
C  Name: WEEKLY                                                          *
C  Module Number: 1.6.1                                                  *
C  Function: Calls PRTHDR to print header HDRNUM; calls RDFMT4 to        *
C            read input file WEEKREQx and write each input record        *
C            to an array incrementing the day number by 7 to create      *
C            11 records.                                                 *
C  Input Parameters: HDRNUM,ICOUNT,FILENM                                *
C  Output Parameters: none                                               *
C  Common Blocks/Variables Used: DIMEN/NUMWKS                            *
C      FLAGS/ECHOFG                                                      *
C      TOTALS/all space                                                  *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: RDFMT4,PRTHDR                                         *
C  Calling Modules: CONREQ                                               *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE WEEKLY(FILENM,HDRNUM,ICOUNT)
      PARAMETER (MAXLOC=3,MAXWKS=11,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      CHARACTER FILENM*7
      DIMENSION SORTAR(1000,5),ISRTAR(1000,5)
      INTEGER  HDRNUM,LOCNUM,INCOMP,CLASS,SIZE,DAYNO,LINES
C
C**** COMMON BLOCKS
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +    NUMAD,NUMOTH,QTRNUM,WPCLAS
      LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/TOTALS/ TOTINT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTBAT(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),
     +    TOTPC(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ)
      EQUIVALENCE (SORTAR(1,1),ISRTAR(1,1),TOTINT(1,1,1,1,1))
C
      IF (ECHOFG) CALL PRTHDR(FILENM,HDRNUM)
```

```
C
   10 CALL RDFMT4(LOCNUM,TIMDAY,INCOMP,CLASS,SIZE,DAYNO)
      IF (LOCNUM .EQ. 0) RETURN
      DO 100  I=1,NUMWKS
          XTIME=(REAL(DAYNO)*86400.)+TIMDAY*3600.
          ICOUNT=ICOUNT+1
          SORTAR(ICOUNT,1)=XTIME
          ISRTAR(ICOUNT,2)=LOCNUM
          ISRTAR(ICOUNT,3)=INCOMP
          ISRTAR(ICOUNT,4)=CLASS
          ISRTAR(ICOUNT,5)=SIZE
          DAYNO=DAYNO+7
  100     CONTINUE
      GO TO 10
      END
C*************************************************************************
C                                                                        *
C  Name: WPWORK                                                          *
C  Module Number: 1.C.18                                                 *
C  Function: Calls PRTHDR to print header HDRNUM; opens input file       *
C            FILENM; reads input file STUDWPx to distribute word         *
C            processing load to TOTAL arrays.                            *
C  Input Parameters: FILENM,HDRNUM                                       *
C  Output Parameters: none                                               *
C  Common Blocks/Variables Used: DIMEN/NUMWKS                            *
C      FLAGS/ECHOFG                                                      *
C      UNITS/NINPUT                                                      *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called:  DISTWP,PRTHDR,RDFMT2,CLSERR                          *
C  Calling Modules: CALSTD,CALADM                                        *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE WPWORK(FILENM,HDRNUM)
      PARAMETER (MAXWKS=11)
C
      CHARACTER FILENM*7
      REAL PERCNT,LDFACT(MAXWKS),INPC
      INTEGER HDRNUM
C
C**** COMMON BLOCKS
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +    NUMAD,NUMOTH,QTRNUM,WPCLAS
      LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
C**** OPEN WORD PROCESSING REQUIREMENTS FILE
      IF (ECHOFG) CALL PRTHDR(FILENM,HDRNUM)
      CALL OPNINP(FILENM,'F')
C
C**** PROCESS FILE
   10 CALL RDFMT2(LOCAT,INSTUD,PERCNT,LDFACT,INPC)
      IF (LOCAT .EQ. 0) THEN
```

```
            CLOSE(NINPUT,ERR=300)
            RETURN
         ENDIF
         INSTUD=INSTUD*PERCNT
C
C**** DISTIBUTE LOAD OVER 11 WEEKS
         DO 100  I=1,NUMWKS
            XTOT=REAL(INSTUD)*LDFACT(I)
            XPC=XTOT*INPC
            XINT=XTOT-XPC
            IWEEK=I
            CALL DISTWP(LOCAT,IWEEK,XINT,XPC)
  100    CONTINUE
         GO TO 10
C
  300    CALL CLSERR(FILENM)
         END
C*************************************************************************
C                                                                        *
C  Name: WRTFIL                                                          *
C  Module Number: 1.C.19                                                 *
C  Function: Calculates and writes the arrival and frequency files       *
C            input parameters.                                           *
C  Input Parameters:  TOTAL,LOCCNT,DAARIV,HRARIV,IFACT                   *
C  Output Parameters: none                                               *
C  Common Blocks/Variables Used: DIMEN/NUMWKS,NUMCPU,NUMSIZ,NUMCLS       *
C      FACTXX/SESFACT,RUNFACT                                            *
C      FLAGS/FRQARV,WRTOUT                                               *
C      NAMES/LOCNMS,COMPNM,CLASNM                                        *
C      UNITS/NPRNT,NOUT                                                  *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: none                                                  *
C  Calling Modules: BATCH,CALSTD,CALADM,CALOTH                           *
C                                                                        *
C*************************************************************************
C
         SUBROUTINE WRTFIL(TOTAL,LOCCNT,DYARIV,HRARIV,IFACT)
         PARAMETER (MAXLOC=3,MAXWKS=11,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
         REAL TOTAL(MAXLOC,MAXWKS,MAXCPU,MAXCLS,MAXSIZ),DYARIV(7),
        +     HRARIV(24),OBSFRQ(MAXSIZ),FREQ(MAXLOC,MAXCPU,MAXCLS,MAXSIZ),
        +     DAYS(7),HOURS(MAXLOC,7,24),NUMSES
C
C**** COMMON BLOCKS
         INTEGER QTRNUM,WPCLAS
         COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
        +     NUMAD,NUMOTH,QTRNUM,WPCLAS
         COMMON/FACTXX/ PCFACT(MAXCPU,MAXSIZ),SESFACT(MAXCPU,MAXSIZ),
        +  RUNFACT(MAXCPU,MAXSIZ),BAFACT(MAXCPU),WPFACT(MAXCPU,MAXSIZ)
         LOGICAL ECHOFG,TABFG,FRQARV,WRTOUT
         COMMON/FLAGS/ ECHOFG,TABFG,FRQARV,WRTOUT
         CHARACTER*8 CLASNM,COMPNM,LOCNMS,SIZNMS,TYPLOC,PERNMS
         COMMON/NAMES/ CLASNM(MAXCLS),COMPNM(MAXCPU),LOCNMS(MAXLOC),
        +     SIZNMS(MAXSIZ),TYPLOC(MAXLOC),PERNMS(10)
         COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
```

```
C
      IF (WRTOUT) WRITE(NOUT) NUMCPU,NUMCLS,NUMSIZ
      DO 900  I=1,NUMWKS
          DO 700  J=1,LOCCNT
              IF (FRQARV) WRITE(NPRNT,910) I,LOCNMS(J)
              TOTSES=0.0
              DO 400  K=1,NUMCPU
                  DO 350  L=1,NUMCLS
                       DO 300  M=1,NUMSIZ
                           IF (IFACT .EQ. 1) THEN
                               NUMSES=TOTAL(J,I,K,L,M)*SESFACT(L,M)
                           ELSE
                               NUMSES=TOTAL(J,I,K,L,M)*RUNFACT(L,M)
                           ENDIF
                           TOTSES=TOTSES+NUMSES
                           FREQ(J,K,L,M)=NUMSES
  300                  CONTINUE
  350              CONTINUE
  400          CONTINUE
               COUNT=0.0
               IF (FRQARV) WRITE(NPRNT,920)
               DO 550  K=1,NUMCPU
                   IF (FRQARV) WRITE(NPRNT,930) COMPNM(K)
                   DO 500  L=1,NUMCLS
                       DO 450  M=1,NUMSIZ
                           IF (TOTSES .GT. 0.0) THEN
                               OBSFRQ(M)=FREQ(J,K,L,M)/TOTSES
                               COUNT=OBSFRQ(M)+COUNT
                           ELSE
                               OBSFRQ(M)=0.0
                           ENDIF
                           FREQ(J,K,L,M)=COUNT
  450                  CONTINUE
                       IF (FRQARV) THEN
                           WRITE(NPRNT,940) CLASNM(L),(OBSFRQ(M),
     +                      M=1,NUMSIZ),(FREQ(J,K,L,M), M=1,NUMSIZ)
                       ENDIF
  500              CONTINUE
  550          CONTINUE
               IF (FRQARV) WRITE(NPRNT,960)
               DO 650  K=1,7
                   DAYS(K)=TOTSES*DYARIV(K)
                   DO 600  L=1,24
                       HOURS(J,K,L)=DAYS(K)*HRARIV(L)
  600              CONTINUE
                   IF (FRQARV) WRITE(NPRNT,970) K,(HOURS(J,K,L), L=1,24)
  650          CONTINUE
  700      CONTINUE
           IF (WRTOUT) THEN
               DO 750 J=1,LOCCNT
                   WRITE(NOUT) (((FREQ(J,K,L,M), M=1,NUMSIZ), L=1,
     +                NUMCLS), K=1,NUMCPU)
  750          CONTINUE
               DO 850 J=1,7
                   DO 800 K=1,LOCCNT
```

```
                    WRITE(NOUT) (HOURS(K,J,L), L=1,24)
  800             CONTINUE
  850           CONTINUE
              ENDIF
  900 CONTINUE
      RETURN
C
C**** FORMAT STATEMENTS
  910 FORMAT('0','WEEK: ',I2,4X,A8)
  920 FORMAT(' ',' DISTRIBUTION FREQUENCY',/,/,
     +    3X,'COMPUTER  CLASS',22X,'OBSERVED FREQUENCY',30X,
     +  'CUMMULATIVE FREQUENCY')
  930 FORMAT('0',2X,A8)
  940 FORMAT('+',12X,A8,4X,5F8.4,10X,5F8.4,//,10X)
  960 FORMAT('0',' ARRIVAL RATES',/,/,
     +  1X,'DAY #      0000   0100   0200   0300   0400   0500   0600',
     +     '   0700   0800   0900   1000   1100',/,8X,'   1200    1300',
     +     '   1400   1500   1600   1700   1800   1900   2000   2100',
     +     '   2200   2300')
  970 FORMAT(' ',2X,I1,6X,12F7.2,/,10X,12F7.2)
      END
C*************************************************************************
C                                                                        *
C  Name: FIGCLS                                                          *
C  Module Number: 1.F.1                                                  *
C  Function: Searchs CLASID array for class ID.  Returns an              *
C            integer postion.                                            *
C  Input Parameters: CLASS                                               *
C  Output Parameters: FIGCLS                                             *
C  Common Blocks/Variables Used: DIMEN/NUMCLS                            *
C      IDS/CLASID                                                        *
C      UNITS/NPRNT                                                       *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: none                                                  *
C  Calling Modules: REGWRK,RDFMT3,RDFMT4                                 *
C                                                                        *
C*************************************************************************
C
      FUNCTION FIGCLS(CLASS)
      PARAMETER (MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      INTEGER FIGCLS
      CHARACTER CLASS*1
C
C**** COMMON BLOCKS
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +    NUMAD,NUMOTH,QTRNUM,WPCLAS
      CHARACTER*1 COMPID,CLASID,SIZIDS
      COMMON/IDS/ COMPID(MAXCPU),CLASID(MAXCLS),SIZIDS(MAXSIZ)
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
      DO 100 I=1,NUMCLS
          IF (CLASS .EQ. CLASID(I)) THEN
              FIGCLS=I
```

```
            RETURN
          ENDIF
  100 CONTINUE
C
C**** PRINT ERROR
      WRITE(NPRNT,200) CLASS
      STOP
  200 FORMAT(' *** ERROR ***  ',A1,' COMPUTER CLASS NOT FOUND')
      END
C*************************************************************************
C                                                                        *
C  Name: FIGCPU                                                          *
C  Module Number: 1.F.2                                                  *
C  Function: Searchs COMPID array for computer ID.  Returns an           *
C            integer postion.                                            *
C  Input Parameters: INCOMP                                              *
C  Output Parameters: FIGCPU                                             *
C  Common Blocks/Variables Used: DIMEN/NUMCPU                            *
C      IDS/COMPID                                                        *
C      UNITS/NPRNT                                                       *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: none                                                  *
C  Calling Modules: REGWRK,RDFMT3,RDFMT4                                 *
C                                                                        *
C*************************************************************************
C
      FUNCTION FIGCPU(AMACH)
      PARAMETER (MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      INTEGER FIGCPU
      CHARACTER AMACH*1
C
C**** COMMON BLOCKS
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +    NUMAD,NUMOTH,QTRNUM,WPCLAS
      CHARACTER*1 COMPID,CLASID,SIZIDS
      COMMON/IDS/ COMPID(MAXCPU),CLASID(MAXCLS),SIZIDS(MAXSIZ)
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
      DO 100 I=1,NUMCPU
          IF (AMACH .EQ. COMPID(I)) THEN
              FIGCPU=I
              RETURN
          ENDIF
  100 CONTINUE
C
C**** PRINT ERROR
      WRITE(NPRNT,200) AMACH
      STOP
  200 FORMAT(' *** ERROR ***  ',A1,' COMPUTER ID NOT FOUND')
      END
C*************************************************************************
C                                                                        *
C  Name: FIGSIZ                                                          *
```

```
C  Module Number: 1.F.3                                                  *
C  Function: Searchs SIZIDS array for size ID.  Returns an              *
C            integer postion.                                           *
C  Input Parameters: SIZE                                               *
C  Output Parameters: FIGSIZ                                            *
C  Common Blocks/Variables Used: DIMEN/NUMSIZ                           *
C      IDS/SIZIDS                                                       *
C      UNITS/NPRNT                                                      *
C  Common Blocks/Variables Changed: none                                *
C  Modules Called: none                                                 *
C  Calling Modules: REGWRK,RDFMT3,RDFMT4                                *
C                                                                       *
C***********************************************************************
C
      FUNCTION FIGSIZ(SIZE)
      PARAMETER (MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
      INTEGER FIGSIZ
      CHARACTER SIZE*1
C
C**** COMMON BLOCKS
      INTEGER QTRNUM,WPCLAS
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMWKS,NUMST,NUMBA,
     +    NUMAD,NUMOTH,QTRNUM,WPCLAS
      CHARACTER*1 COMPID,CLASID,SIZIDS
      COMMON/IDS/ COMPID(MAXCPU),CLASID(MAXCLS),SIZIDS(MAXSIZ)
      COMMON/UNITS/ NCRDR,NPRNT,NINPUT,NOUT
C
      DO 100 I=1,NUMSIZ
          IF (SIZE .EQ. SIZIDS(I)) THEN
              FIGSIZ=I
              RETURN
          ENDIF
  100 CONTINUE
C
C**** PRINT ERROR
      WRITE(NPRNT,200) SIZE
      STOP
  200 FORMAT(' *** ERROR ***  ',A1,' COMPUTER SIZE NOT FOUND')
      END
```

APPENDIX E

NETWORK MODEL USER'S GUIDE

Introduction

The AFIT network simulation model performs a discrete simulation of the AFIT ADP network. This simulation may encompass an entire school quarter or portion of week(s) within the quarter. The hardware configuration is defined by the use of SLAM network statements and an input factor file. The user arrivals and resource demands for each resource location of the network are provided by input files created by the AFIT workload model.

The model was developed for the AFIT ADP network system; however, it may be easily configured for virtually any ADP network. All of the network resources (computers, terminals, card readers, and printers) are defined by SLAM network statements and an input file. The only software module requiring modifications for a configuration change is Function FIGRUN which returns run times for each computer resource attached. This user's guide reflects the current AFIT configuration. Table E.1 lists the resources and their mnemonics used throughout this guide. Computer resources and their ports are defined by global variables as described later.

The model is driven by the arrival rates and cumulative frequency distributions provided for each input location (interactive and batch) of the network. These files are created by the AFIT workload model. Therefore, the network model must be configured identically to the workload model. Please reference the AFIT Workload Model User's Guide for a description of the workload model configuration.

Mnemonic	Description
Terminals:	
STUD125	Student Building 125
STUD640	Student Building 640
STUD641	Student Building 641
FAAD125	Faculty/Administrative Building 125
FAAD640	Faculty/Administrative Building 640
FAAD641	Faculty/Administrative Building 641
SOFT125	Software Development Building 125
Card Readers:	
RBT125CR	Remote Batch Station Building 125
HAR80CR	Harris 80 Building 640
RBT641CR	Remote Batch Station Building 641
HAR500CR	Harris 500 Building 641
Printers:	
RBT125LP	Building 125 Line Printer
RBT641LP	Building 641 Line Printer
HAR80PEN	Harris 80 Pen Plotter
HAR80LP	Harris 80 Line Printer
HAR80PP	Harris 80 Printer/Plotter
SSCPP	VAX 11/780 Printer/Plotter
HAR500LP	Harris 500 Line Printer

Table E.1 Resource Definitions.

This user's guide used in conjunction with the Network Model Maintenance Guide provides all the data required to operate the model. Descriptions of all the input files are provided. The input variables description and format are provided. Creation and modification of the SLAM network structure is reviewed. Lastly, the CDC CYBER job control inputs, error conditions, and limitations of the model are reviewed.

Inputs

Input files to the AFIT network model consist of the five files created by the workload model, a factor file to

File Name	Description
STUDxxy	Student Arrival Rates and Frequency Distributions
ADMNxxy	Faculty/Administrative Arrival Rates and Frequency Distributions
OTHRxxy	Software Development Arrival Rates and Frequency Distributions
BTCHxxy	Batch Arrival Rates and Frequency Distributions
CNSTxxy	Periodic Requirements

xx - quarter ID
y - unique alpha ID

Table E.2 Network Model Input Files.

initialize variables, and namelist inputs (Table E.2). The file names of the files created by the workload model consist of seven characters. The first four characters are constant. The next two characters identify the quarter (WI,SP,SU, or FA) and the last character is an unique identifier provided by the user of the workload model. The factor file's name is user defined.

Workload Model Files. Five files are created by the workload model to drive the network model. Four files consist of arrival rates and cumulative frequency distributions for all network input locations. The last file defines periodic requirements such as scheduling programs which could not be included in the other file's generation. These periodic requirements are routed to the input locations as required. Please reference the Network Model Maintenance Guide if the input locations are modified.

Factor File. A factor file must be created to initialize: the mean and standard deviation of various distributions required to drive the simulation; probability

```
 Format          Description = Variable Names

namelist          &LDIMEN
                  **** ANY CODES ****
8X,12F6.4         CLS  X     (ANYDST)
                  **** 'BATCH' PROBABILITIES IN INTERACTIVE ***
8X,12F6.4         CPU  X     (BATINT)
                  **** DIALUP PROBABILITIES ****
8X,12F6.4         CYBER      (DIALUP)
                  **** DISTRIBUTION TABLE ****
8X,4F5.1,         CLS SIZ    (DISTRB(1-5))
2F10.4,4F5.1
8X,2F10.4                    (DISTRB(6))
                  **** WORD PROCESSING PROBABILITIES ****
8X,12F6.4         ONEJOB     (ONEJOB)
8X,12F6.4         CLS A-F    (WPPROB)
                  **** PRINTER SPEEDS ****
8X,F8.0           PRINTER    (PRTSPD)
                  **** PRINTER PROBABILITIES ****
8X,12F6.4         DIALUP       (DUPRT)
8X,12F6.4         PRT LOC      (PRTTAB)
                  **** CPU INITIALIZATION UP/DOWN, MPL, ....
8X,3F8.0,F8.4     COMPUTER     (flag,mpl,ports,CNVCPU)
                  **** COMPUTER SCHEDULES ****
8X,I8             COMPUTER     (sequence number)
list                 (SCHCPU)
8X,I8             STOP      0
                  **** BATCH LOCATION SCHEDULES ****
8X,I8             CARD RDR     (sequence number)
list                 (SCHBAT)
8X,I8             STOP      0
                  **** TERMINAL WAIT PROBABILITIES BY LOC ****
8X,12F6.4         LOC            (TRMWAT)
                  **** PRINTERS ASSOCIATED WITH GATES ****
namelist          &PRTGAT     (IPRGAT)
                  **** CARD READERS ATTACHED TO COMPUTERS ****
namelist          &CARDGAT    (ICRGAT)
```

Table E.3 Network Factor File Format.

mass functions when a routing decision needs to be made; and various tables to define the complex relationships between computers, locations, and so forth. The factor file format is illustrated in Table E.3 and an example is provided in Table E.4. Further descriptions of the variables may be found to the Network Model Data Dictionary (Appendix G).

```
 &LDIMEN NUMLOC=10,NUMCPU=4,NUMCLS=6,NUMSIZ=5,NUMPRT=7,NUMGAT=4,
   IBATST=8,IWPCLS=6 &END
XXX ANY CODES XXX
CLS A   0.25  0.50  0.75  1.00
 .  .    .     .     .     .
XXX 'BATCH' PROBABILITIES IN INTERACTIVE XXX
CPU 1   0.50  0.50  0.50  0.50  0.50  0.50
 .  .    .     .     .     .     .     .
XXX DIALUP PROBABILITIES XXX
CYBER   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.10  0.10  0.05  0.05  0.05
        0.05  0.05  0.05  0.05  0.15  0.20  0.25  0.25  0.30  0.30  0.35  0.30
  .      .     .     .     .     .     .     .     .     .     .     .     .
         .     .     .     .     .     .     .     .     .     .     .     .
XXX DISTRIBUTION TABLE XXX
 A 1      600  100    5    1    2.1       0.5       5    2  100  100
           50.0       10.0
 . .       .    .     .    .    .         .         .    .  .    .
                .          .
 XXX WORD PROCESSING SWITCH(PRNT) PROBABILITIES XXX
ONEJOB  0.3
CLS A-F 0.25  0.20  0.15  0.10  0.05  0.05
XXX PRINTER SPEEDS XXX
RBT125LP     1000
   .          .
XXX PRINTER PROBABILITIES XXXX
DIALUP  0.25
  1 1   1.00  1.00  1.00  1.00  1.00  1.00  1.00
    2   0.00  1.00  1.00  1.00  1.00  1.00  1.00
    .    .     .     .     .     .     .     .
   10   0.00  0.00  0.25  0.75  1.00  1.00  1.00
  . .    .     .     .     .     .     .     .
XXX CPU INITIALIZATION UP/DOWN, MPL, # PORTS, CM/CPU XXXX
CYBER         0.     10.     50.  1.0
  .           .      .       .   .
XXXX COMPUTER SCHEDULES XXXX
CYBER         1
  25200.  25200.  25200.  25200.  25200.  25200.  25200.
  85800.  85800.  85800.  85800.  85800.  85800.  85800.
STOP          0
XXXX BATCH LOCATION SCHEDULES XXXX
RBT125CR      1
  28800.  28800.  28800.  28800.  28800.     -1.     -1.
  59400.  59400.  59400.  59400.  59400.     -1.     -1.
STOP          0
XXXX TERMINAL WAIT PROBABILITIES BY LOCATION XXXX
LOC 1   0.8   0.5   0.4   0.2   0.1
 .  .    .     .     .     .     .
XXXX PRINTERS ASSOCIATED WITH GATES - IPRGAT(GATE,PRINTER) XXXX
 &PRTGAT IPRGAT(1,1)=1,IPRGAT(2,3)=1,IPRGAT(2,4)=1 &END
XXXX CARD READERS ATTACHED TO COMPUTERS - ICRGAT(CPU,GATE) XXX
 &CARDGAT ICRGAT(1,2)=1,ICRGAT(3,4)=1 &END
```

Table E.4 Network Factor File Example.

Variable	Description
NUMLOC	Number of Input Locations
NUMCPU	Number of Computers
NUMCLS	Number of Classes
NUMSIZ	Number of Sizes
NUMPRT	Number of Printers
NUMGAT	Number of Gates (Card Readers)
IBATST	Batch Location Start Number
IWPCLS	Word Processing Class Number

Table E.5 LDIMEN Namelist Variables.

The first record is a FORTRAN namelist statement used to set the dimensions of the model. These variable names and descriptions are listed in Table E.5. The next record defines a cumulative probability mass function for each class (ANYDST). This function is used to assign a computer to an arrival when the 'any' computer has been determined by the workload cumulative frequency distribution. 'Batch' probabilities in interactive (BATINT) define the probability of the user submitting a job in 'wait mode' (will wait for completion) or 'batch mode'. These probabilities are defined for each computer and size work.

DIALUP defines the probability mass function that an interactive arrival originates at an external (dial-up) input terminal. Different functions are provided for each computer. The distribution table (DISTRB) defines a mean and standard deviation for the think time between job submittals, number of K lines to print, CPU seconds required, expected number of jobs per session, memory size required, and I/O seconds required. These values are provided for

each class and size defined. Word processing variables are provided next. ONEJOB is the probability that the last job submitted by an interactive user will require a printer. The next record defines WPPROB which is the probability for each class that a printer is required.

The printer's characteristics are defined next. The printers' speeds (lines/minute) are defined in array PRTSPD. A record is required for each printer in the order they appear in the network structure. The possible selection of a personal computer as a printer is defined by DUPRT. The next records define, for each computer and input location, a cumulative probability mass function which defines the printers accessable from the computers.

Each computer's initialization is set by four values. The first value indicates whether the computer is operational (1) or not operational (0). The next value sets the computer's multiprogram level. The third value sets the number of ports available for terminals. Lastly, the CPU conversion factor is set. This factor is used to convert CPU time from one computer to another. For example, if the CYBER's factor equals 1.0, a job on the CYBER takes 60 seconds, and the same job takes 30 seconds on the VAX, then the VAX conversion factor would be 0.5.

The operational schedules of each computer are provided next. If a computer's schedule is provided, the computer's sequence number is input followed by the operational up times for each day followed by the operational down times for each day. After the desired schedules are input the

Variable	Default	Description
ISUM	1	Summary Flag
INTER	6	Statistics Collection Interval
IWEEK	1	Start Week Number
ISTOP	11	Stop Week Number
XLOGON	0.0	Log On Time
XLOGOF	0.0	Log Off Time
LOGNL	none	LOGFLAG Namelist Flag
VARNL	none	VARTIME Namelist Flag

Table E.6 CONFIG Namelist Variables.

"STOP 0" record must be included. The operational schedules for each card reader (gate location) is provided in the same mannner.

The terminal wait probabilities for each terminal input location may be input. A maximum of parameter MAXWAT for each location may be input (see maintenance manual). The relationships between the printers and card reader locations are defined by namelist input PRTGAT. For every printer linked to a card reader (same operational schedule), the array IPRGAT(gate number,printer number) must be set to 1. Lastly, the relationships between the card readers and computers are defined in the same manner using namelist CARDGAT. For every card reader linked to a computer (computer not operational - card reader not operational), the array ICRGAT(computer number, gate number) is set to 1.

Namelist Inputs. Three namelist inputs may be input to change the model's default values. The first, CONFIG, must be input. The variables set by this statement and their default values are listed in Table E.6. ISUM defines

Log	Description	Log	Description
1	ARVL-UPDAT	13	RTPRT
2	ARVL-SUBMT	14	SETINT
3	CPUARV-HLD	15	WAITTR-OK
4	CPUARV-RUN	16	WAITTR-LV
5	CPUDPT	17	INTHND
6	CPUDPT-NEW	18	RESORC
7	SESOVR	19	ENTINT
8	LATARV	20	SETCPU
9	LOGON	21	STPINT
10	PERIOD-GO	22	WAITPO-OK
11	PERIOD-SCH	23	WAITPO-LV
12	RESCHD		

Table E.7 Network Logs.

whether the SLAM Summary Reports are generated. INTER is the statistics collection interval. Every INTER hours a SLAM Summary Report is output and the statistical arrays are cleared. IWEEK and ISTOP define the simulation run's start and stop week numbers respectfully. XLOGON and XLOGOF define the times the logging function is activated and deactivated. Lastly, LOGNL and VARNL must be set to .TRUE. or .FALSE.. If LOGNL is true, namelist LOGFLAG follows. If VARNL, is true, namelist VARTIME follows.

Namelist LOGFLAG is used to define the logs which will be output between XLOGOF and XLOGON time. A log consists of TNOW, description, and attribute values of an event. A list of the log numbers and descriptions is provided in Table E.7. If log number 6 is desired, then LOGON(6)=1 would be included in the namelist input. All of the logs are defaulted off (0). Further information on logging is provided in the maintenance guide.

Variables		
Mean	SID	Description
CPULTM	CPULTS	Computer Not Available Reschedule Time
TRMLTM	TRMLTS	Terminal Not Available Reschedule Time
OPRMN	OPRSTD	Batch Operator Input Time
XLOGMN	XLOGST	Interactive Log On Time
WAITMN	WAITST	Computer Port Not Available Reschedule Time
PCPRTM	PCPRTS	Personal Computer Print Time

Table E.8 VARTIME Namelist Variables.

Namelist VARTIME is used to change any of the means and standard deviations of various time delays as listed in Table E.8. All of the values are defaulted by the BLOCK DATA module. Please reference BLOCK DATA for these defaults. Further descriptions of these variables may be found in the data dictionary.

An example of the namelist inputs is:

```
&CONFIG INTER=12, XLOGON=0.0, XLOGOF=86400.,
     VARNL=.TRUE., LOCNL=.TRUE. &END
&LOGFLAG LOGON(6)=1, LOGON(15)=1 &END
&VARTIME OPRNM=100. &END
```

In this example the SLAM Summary Report would be printed and the statistical arrays cleared every 12 hours (INTER=12). Logs would be output for the first day (XLOGON=0.0, XLOGOF=86400.). Namelist LOGFLAG and VARTIME will follow. Log 6 and 15 would be output during the first day. Finally, the batch operator input mean would be set to 100 seconds.

Outputs

The only output generated by the network model is the SLAM Summary Report (Ref 20:57) followed by the CPU and I/O utilizations for each computer. Additionally, the logs as described earlier may be output.

Network Structure

As stated earlier, the hardware configuration of the AFIT network is structured using SLAM network statements. Familiarity with SLAM is assumed for this discussion. Also please reference the network listing in Appendix H.

Control Statements. For the LIMIT's statement, a minimum of 40 plus the number of computers files (MFIL) and 15 attribute values (MATR) must be declared. TIMST statements are used to collect time persistent statistics of the computers availability and number of ports available. These values are maintained in the XX array. The first availability value for the computers is XX(10). The other computers follow sequentially. This sequence must match the sequence of computer definitions in the input factor file. The port availabilities are maintained in the same manner starting at XX(40). STAT statements must be included to collect each computer resource response time. Again, the same sequential order is maintained. The INIT statement defines a simulation run time of 0.0 to 6800000. seconds (11 weeks).

Resource and Gate Blocks. The interactive input locations are defined first by using SLAM resource blocks starting with file number 10. The capacity is set to the number

of terminals available at each location. The printers are also defined by resource blocks starting with file number 20. The capacity is either 0 or 1 depending on the printer's status at start up (2400 Sunday). If the printer is operational, set the capacity to 1. Otherwise, set the capacity to 0.

The batch locations are defined using SLAM gate blocks starting at file number 30. The status is set to close or open depending on the location's status at start up. Note that four batch locations are defined but only three batch locations were defined using the workload model. RBT641CR and HAR500CR were considered one location by the workload model. This relationship will be discussed further in the batch arrivals section.

Interactive Arrivals. Terminal resources are seized by the model in this section. The discrete structure of the model enters an event through node 1 with ATRIB(7) equal to the input location number. This number matchs the sequence defined for the terminal resource blocks above except 0 which is used to define a dial-up arrival. A jump (ACT) statement must be included for every terminal location. The event takes the appropriate jump to seize the terminal (AWAIT). For all locations, a jump to LOGON is then taken and after a log on time (USERF(1)) the event is routed back to the discrete structure.

Batch Arrivals. Card reader (batch) resources are defined similar to the terminal resources. The discrete structure of the model enters an event through node 2 with

ATRIB(7) equal to the input location number. This number begins with the next integer number available after the terminal resources. For example, the last terminal resource defined is SOFT125 (number 7); therefore, the first batch location number is 8. Again, a jump statement is provided for every batch location. If the input 'building' is 641 then another jump is taken to either RBT641CR or HAR500CR. Eacn event is then processed by an AWAIT node, operator input time (USERF(2)), the wait time is collected, and then routed back to the discrete structure.

Interactive Release. After a session is completed, an event is entered at node 3 to release the terminal resource. A jump table follows which is identical to the Interactive Arrivals table. The appropriate jump is taken to FREE the terminal, collect the session time, and terminate the event.

Printers. All requests for printers are entered through node 4 with ATRIB(15) equal to the printer number. The printer number corresponds to the sequence of resource block definitions and the printer definitions in the input factor file. The last jump in the jump table is added for a personal computer print. After the event takes the appropriate jump, the event waits for the printer at the AWAIT node. ATRIB(8) contains the print time. After the resource is seized and 'printing' completes, the resource is released (FREE). The printer time and the printer plus queue time (INT(6)) are collected. If this is a batch job (ATRIB(14) > 7), the event is routed to BACL to collect statistics.

Collect Batch Statistics. At BACL optional turnaround times may be collected. ATRIB(7) contains the computer number and ATRIB(14) contains the location number. For this development, turnaround times by computer and building were collected.

Job Setup

The model currently runs on the ASD CYBER computer. The job may be entered through interactive batch or regular batch input. Typical job statements are:

```
MCD,T70,IO170,CM240000.  T830499,MCDERMOTT,55533
ATTACH,SIMS,SIMS.
ATTACH,LGO,SIMLGO.
ATTACH,TAPE8,CONSLAM.
ATTACH,TAPE9,STUDWIC.
ATTACH,TAPE10,ADMNWIC.
ATTACH,TAPE11,OTHRWIC.
ATTACH,TAPE12,BTCHWIC.
ATTACH,PROC,SLAMPROC,ID=AFIT,SN=AFIT.
BEGIN,SLAMII,PROC,M=LGO,PMD=1,I=SIMS).
```

Error Conditions

Minimum error checking is provided by the model. The model's value is totally reflected by the accuracy of the network structure and the factor file. Careful attention should be paid to these files creation. The only error conditions detected by the model are: 1 - Invalid USERF call from network and 2 - Input file mismatch with input CPU, class, or size dimensions.

Limitations

The network model capacity is currently limited by the SLAM file structure and the FORTRAN parameter statements in the discrete structure. Currently, the SLAM file

definitions limit the terminal locations, printers, and batch locations to ten. The computers are limited to five. The parameter statements limit the total input locations to ten; the classes and sizes to six; and the number of gates and wait probabilities to four.

APPENDIX F

NETWORK MODEL MAINTENANCE GUIDE

Introduction

The AFIT Network Model is a top-down designed simulation of the AFIT ADP network using SLAM. It utilizes SLAM's combined network-discrete structure. The 'network' structure is reviewed in the AFIT Network Model User's Guide. This appendix reviews the 'discrete' structure of the network model. The discrete structure is written in FORTRAN Version 5 and is currently running on ASD's CYBER computer in batch mode.

The objective of this guide is to review the documentation standards, programming standards, and provide the general structure of the AFIT Network Model. The documentation standards are very similar to the standards used by the workload model. The programming standards were primarily developed to utilize SLAM common blocks and document the discrete structure with its interface to the network structure. Please reference the AFIT Network Model User's Guide (Appendix E) for a description of the network structure.

Documentation Standards

The bulk of the detailed documentation on the AFIT network model resides within the coded modules. A specific header was used for this documentation. This standard calls for the following header information on each module:

```
C  Name:
C  Module Number:
C  Function:
C  Attributes Referenced:
C  Global (XX) Variables Referenced:
C  Common Blocks/Variables Used:
```

```
C  Common Blocks/Variables Changed:
C  Modules Called:
C  Calling Modules:
C  Scheduled by:
```

Besides the header information, comment statements for each module are included within the body of the module.

Each module's structure followed a specific standard. All DO loops and IF THEN ELSE bodies were indented. FORTRAN statements for each module followed the following order:

SUBROUTINE or FUNCTION statement
PARAMETER statements
Local declarations
COMMON statements
EQUIVALENCE statements
The body
RETURN statement
FORMAT statements
END statement

Also included within the model is a BLOCK DATA module. This module includes every common block and sets all default values except the global (XX) variables.

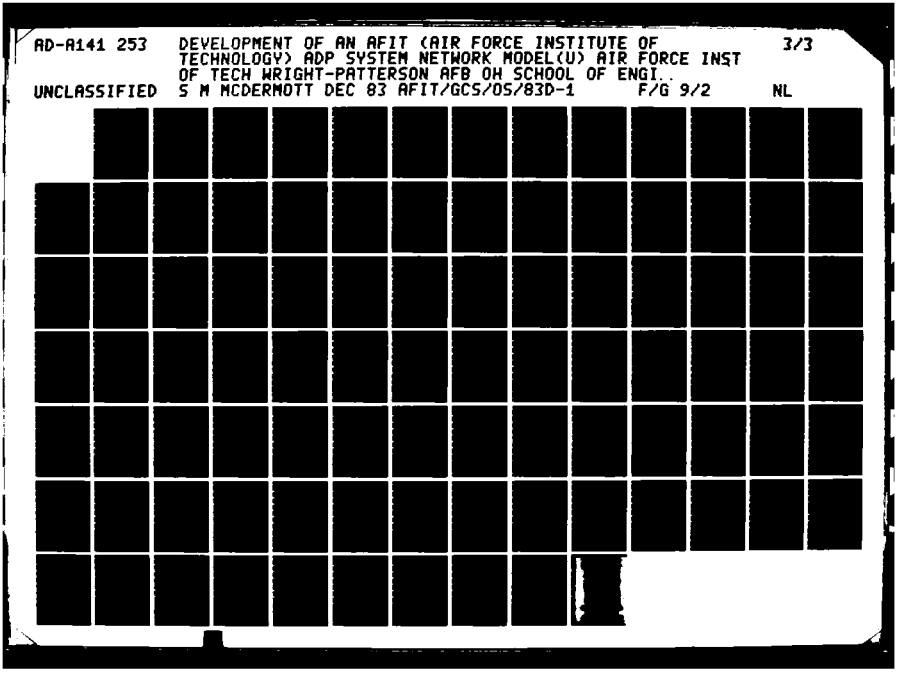
The data dictionary is included in Appendix G. This dictionary also followed a standard format as follows:

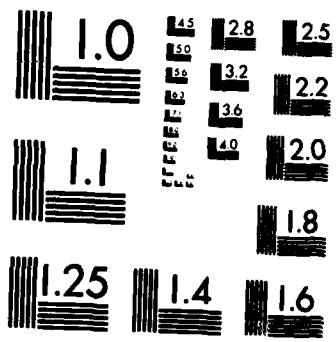
Name: Type:
Common Block:
Description:
Initialized by:
Used by:

The items in the data dictionary include all common variables and names for PARAMETER constants.

Programming Standards

In order to establish the discrete structure's interface to the SLAM executive and the network structure, programming standards were developed. These standards allowed the utilization of the SLAM common variables thereby


AD-A141 253 DEVELOPMENT OF AN AFIT (AIR FORCE INSTITUTE OF TECHNOLOGY) ADP SYSTEM NETWORK MODEL(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI..
3/3
UNCLASSIFIED S M MCDERMOTT DEC 83 AFIT/GCS/OS/83D-1 F/G 9/2 NL



Number	Mnemonic	Description
1	ATIME	Arrival Time
2	ILOCAT	Input Location
3	IMACH	Computer
4	ICLASS	Class
5	ISIZE	Size
6	APRTQ	Input Time to Card Reader or Print Queue
	AWAIT	Wait Time for Location Change
7	ACPSEC	CPU Seconds
	ALOCAT	Location Jump for Network
8	ARUN/APRTL	Run Time / Printer Lines
	ACPU	CPU Jump for Network
9	INTBAT	Interactive State Flag 0/1 = Batch/Wait Mode
10	IJOBS	Number of Jobs This Session
11	IDIAL	Dial-up Flag 0/1 = No/Yes
12	IORG	Original Class
13	AMEAN	Mean Time of Next Job Submittal
14	AMEM	Memory Required
15	AJUMP	Jump Flag for Network

Table F.1 Attribute Mnemonics and Descriptions.

reducing memory requirements and increased the versatility of the discrete structure. The discrete structure will not require modification as the hardware configuration (network structure) is modified. Additionally, these standards enhance the readability of the modules and provide increased documentation.

Number	Mnemonic	Network Location	Subroutine Referenced
1	INTARV	Interactive Arrivals	INITLC,ARVL, PERIOD
2	IBATCH	Batch Arrivals	INITLC,ARVL, PERIOD
3	NRLSIN	Interactive Release	LOGON,SESOVR
4	IPRTEN	Printers	RTPRT

Table F.2 Enter Node Descriptions.

Event Attributes. SLAM uses a common array (ATRIB) in the SCOM1 common block to store event attribute values. Equivalence statements are included in every module to reference these attributes as listed in Table F.1. Besides increasing the readability of the code, this standard allows the actual attribute numbers to change without changing the code. To change an attribute number only the equivalence statements need to be modified.

Network-Discrete Interface. SLAM allows combined network-discrete simulation by use of enter and event nodes (network) and ENTER and EVENT subroutines (discrete). To enter an event to the network structure, the mnemonics defined in Table F.2 are used. To schedule an event on the event calendar, the mnemonics defined in Table F.3 are used. Three of the locations are scheduled by the network structure. Again, this enhanced the readability of the code and only requires minimum effort to change actual event or enter numbers.

Number	Subroutine Location	Network Location	Mnemonic
1	CLOCK		ICLKEV
2	INTHND	Printers	INTHEV
3	RESORC		IRESEV
4	SETCPU		ISTCEV
5	ARVL		IARVEV
6	PERIOD		IPEREV
7	LOGON	Interactive Arrivals	
8	CPUDPT		ICPDEV
9	CPUARV	Batch Arrivals	
10	LATARV		LATEVT

Table F.3 Event Numbers Descriptions.

Global(XX) and State(SS) Variables. The arrays XX and SS defined in the SLAM SCOM1 common block are used to store global variables and the computer state respectfully. Like the event attributes, FORTRAN equivalence statements are used to reference these variables. Table F.4 lists the XX global variables. Further description of the variable mnemonics may be found in the data dictionary. The XX array is also used to track the computer availabilities starting at number 10 and the computer ports starting at number 40. These values are referenced by adding an offset variable to the computer sequence number. For the computer availability, the offset (CPUOS) equals 9. For the ports, the offset (IPORTS) equals 39.

XX()	Mnemonic	Description
1	IHOUR	Hour Number.
2	IDAY	Day Number.
3	IWEEK	Week Number.
4	IWPCLS	Word Processing Class.
5	UPTTIM	Time of Next Update.
6	ISTOP	Stop Week Number.
7	XLOGON	Log On Time.
8	XLOGOF	Log Off Time.
9		Unused.
10		CYBER Status.
11		CREATE Status.
12		Harris 500 Status.
13		VAX 11/780 Status.
14-16		Unused.
17	ISUM	Summary Print Flag.
18		Unused.
19	INTER	Statistic Collection Interval.
20	ITOTAL	Statistic collection counter.
21-39		Unused.
40		CYBERS Ports.
41		CREATE Ports.
42		Harris 500 Ports.
43		VAX 11/780 Ports.
44-100		Unused.

Table F.4 Global (XX) Variables.

SS(INDEX+)	Mnemonic	Description	Init
1	ITRMOL	# Terminals Online	0.0
2	ITRMWP	# Terminals Word Processing	0.0
3	ITRMAC	# Terminals Active	0.0
4	IEXJOB	# Jobs Running	0.0
5	IMEM	Total Memory Req	0.0
6	ICPU	CPU Seconds Used	0.0
7	INOUT	I/O Seconds Used	0.0
8	IMPL	Multiprogram Level	RDPARM

Table F.5 Computer State Variables.

The computer states are maintained in the SS array as listed in Table F.5. Each state is referenced by calculating the index and then adding the appropriate mnemonic. This index (INDEX) is calculated by multiplying the computer sequence number (IMACH) by 10 and then subtracting 10 as follows:

INDEX = IMACH * 10 - 10

Files. The files used by the simulation are defined in Table F.6. Access to the file data by the discrete structure is achieved by adding a file offset to the computer sequence number or input location sequence number. The files 1 through 9 are used to store events in the CPU 'input hold queue'. The files are referenced by adding the offset (IHLDOS) to the computer number (IMACH). Files 10 through 19 are used to store events waiting for a terminal.

File	Description	File	Description
1	CYBER HOLD	22	HAR80PEN
2	CREATE HOLD	23	HAR80LP
3	HARRIS 500 HOLD	24	HAR80PP
4	SSC HOLD	25	SSCPP
5-9	Unused	26	HAR500LP
10	STUD125	27-29	Unused
11	STUD640	30	RBT125CR
12	STUD641	31	HAR80CR
13	FAAD125	32	RBT641CR
14	FAAD640	33	HAR500CR
15	FAAD641	34-39	Unused
16	SOFT125	40	CYBER Ports
17-19	Unused	41	CREATE Ports
20	RBT125LP	42	Harris Ports
21	RBT641LP	43	VAX 11/780 Ports
22	HAR80PEN		

Table F.6 Network File Descriptions.

The files are reference by adding the offset (ITRMOS) to the input location number (ILOCAT). Files 20 through 29 (printers) and files 30 through 39 (card readers) are not referenced by the discrete structure. Lastly, files 40 through 43 are used to store events waiting for a computer port. These files are referenced by adding the offset (IPORTS) to the computer number (IMACH).

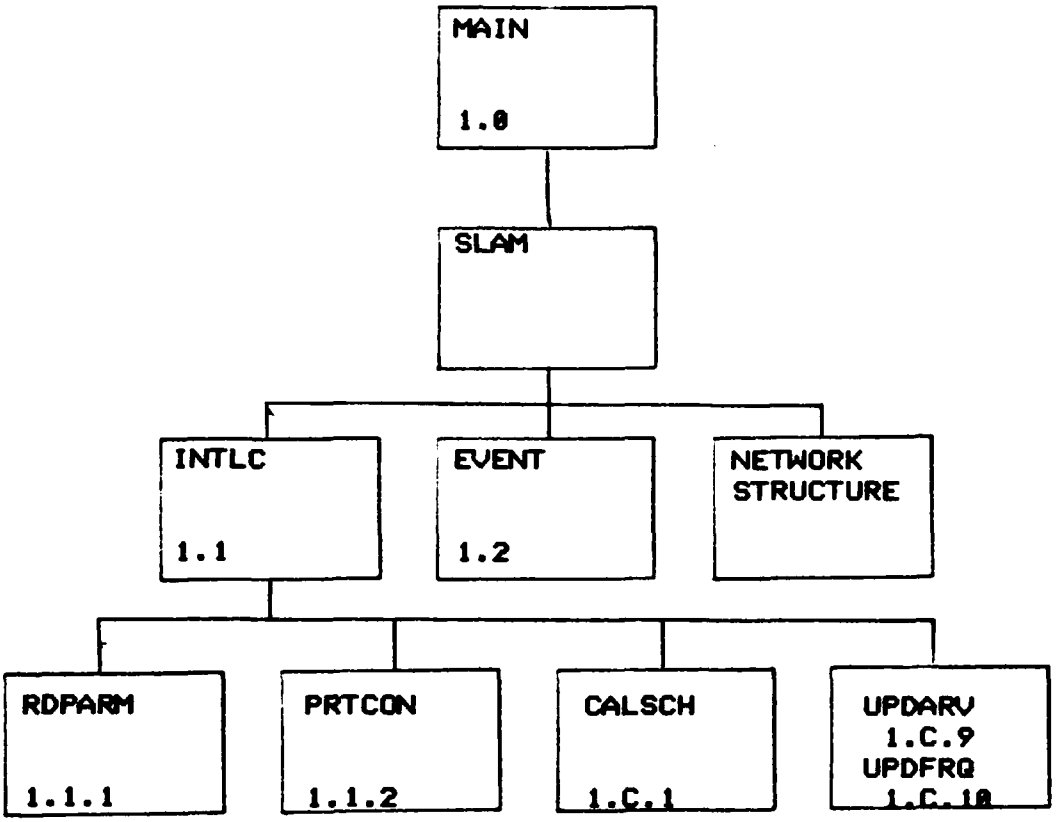


Figure F.1 Network Model (MAIN) Hierarchy.

General Structure

As previously mentioned, the AFIT Network Model has a top-down structure. At the top of this structure is the MAIN module as illustrated in Figure F.1. Table F.7 lists all the discrete modules with a short description. After the MAIN level, the SLAM executive processes all events through the EVENT module. All called modules in the EVENT module are numbered 1.E.x where x is the sequence in the module. Each of these modules are presented in this section.

Module Name	Module Number	Page Number	Description
ARVL	1.E.5	I-2	Handles all network arrivals.
BLOCK	1.B	I-3	BLOCK DATA
CALSCH	1.C.1	I-5	Schedules resources up/down times.
CLOCK	1.E.1	I-5	Performs all time processing.
CPUARV	1.E.9	I-6	Handles all CPU arrivals.
CPUDPT	1.E.8	I-8	Handles all CPU departures.
DYPROC	1.E.1.2	I-9	Performs end of day processing.
EDPROC	1.E.1.4	I-9	Performs end of run processing.
ENTINT	1.C.2	I-10	Checks resource availability for an interactive arrival.
EVENT	1.2	I-11	Calls appropriate subroutines.
FIGUTL	1.E.1.1 .1	I-12	Figures and prints CPU and I/O utilizations.
GETARV	1.E.5.1	I-13	Sets arrival attributes.
HRPROC	1.E.1.1	I-14	Performs end of hour processing.
INTHND	1.E.2	I-15	Handles all interactive sessions.
INTLC	1.1	I-16	Initializes all variables.
LATARV	1.E.10	I-19	Enters rescheduled events into network.
LOG	1.C.3	I-19	Logs subroutine name and attributes.
LOGERR	1.C.4	I-20	Logs user errors.
LOGON	1.E.7	I-20	Checks port availability.
PERIOD	1.E.6	I-21	Handles all periodic arrivals.
PRTCON	1.1.2	I-23	Prints the variables.
RDERR	1.C.5	I-26	Handles file read errors.
RDPARM	1.1.1	I-27	Reads the input factor file.
RESCHD	1.C.2.1	I-30	Reschedules arrivals.
RESORC	1.E.3	I-31	Resets status of resources.
RTPRT	1.E.8.1	I-32	Determines printer and routes to network.
SESOVR	1.C.6	I-33	Performs end of session processing.
SETCPU	1.E.4	I-34	Sets CPU states and session time.
SETINT	1.C.7	I-35	Sets interactive job's print and mode characteristics.
STPINT	1.E.3.1	I-36	Released all interactive users.
UPDARV	1.C.8	I-37	Updates the arrival rates.
UPDFRQ	1.C.9	I-38	Updates the frequency distributions.
WKPROC	1.E.1.3	I-39	Performs end of week processing.
ARVTIM	1.F.1	I-40	Returns next arrival time.
CPUSEC	1.F.2	I-40	Returns CPU seconds required.
FIGRUN	1.F.3	I-41	Returns run time required.
MEMSIZ	1.F.4	I-41	Returns memory required.
PRTLIN	1.F.5	I-42	Returns print lines required.
USERF	1.F.6	I-42	Returns time for network structure.
WAITPO	1.F.7	I-43	Determines if interactive arrival will wait for port.
WAITTR	1.F.8	I-44	Determines if interactive arrival will wait for terminal.
XIOSEC	1.F.9	I-45	Returns I/O seconds required.

Table F.7 Network Modules Descriptions.

The MAIN module is exactly as Prisker defines (Ref 20:232) and is primarily used to set the NSET/QSET dimension. The SLAM executive is called which calls INTLC. INTLC defaults the run configuration as defined in the user's guide, calls RDPARM to input the factor file, and calls PRTCON to output the factor file values. CALSCH is then called to schedule each computer's and batch location's operational up and down times. The arrival rates (ARRATE) and cumulative frequency distributions (FREQ) are initialized by calling modules UPDARV and UPDFRQ respectfully. The arrival rates are used to schedule the first input locations (interactive and batch) arrivals. Finally, the CNSTxxy input file is read to schedule the first periodic arrival.

The first event module is CLOCK as illustrated in Figure F.2. CLOCK controls all of the time processing. First, it updates the update time (UPTTIM) and reschedules itself for an hour later. HRPROC is called to check the statistic interval (INTER). If this interval has been reached, the SLAM Summary Report is output, FIGUTL is called to calculate and output the CPU and I/O utilizations, and the statistical arrays are cleared. If the end of day has been reached, CALSCH is called to schedule the computers' and card readers' changes in operational status. DYPROC also updates the arrival rates if the end of the week has not been reached. If the end of week has been reached, WKPROC is called to update the arrival rates and cumulative frequency distributions for every location. Finally, if the

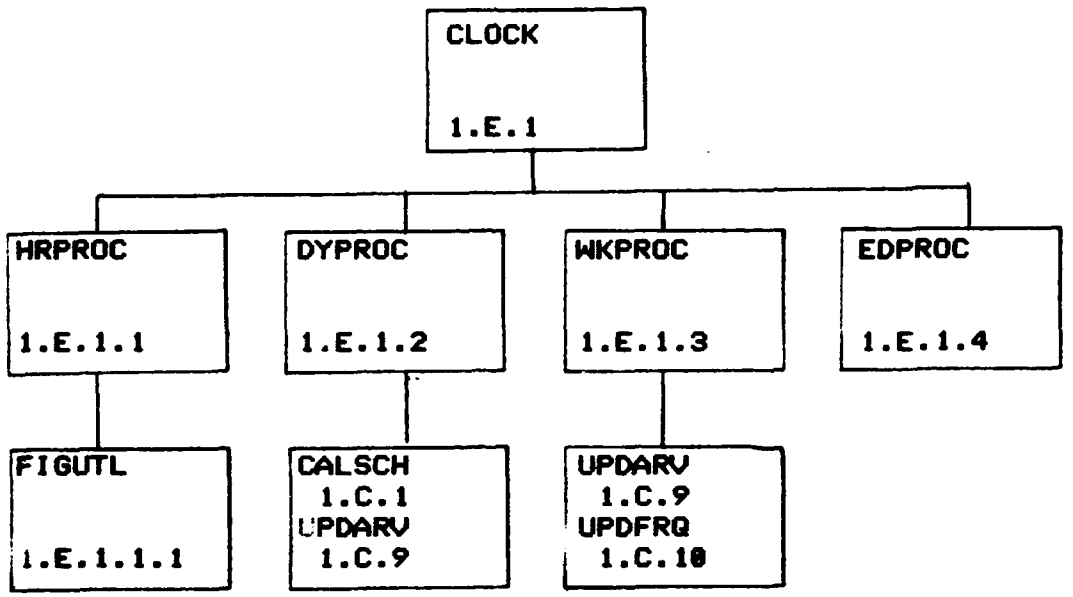


Figure F.2 CLOCK Hierarchy.

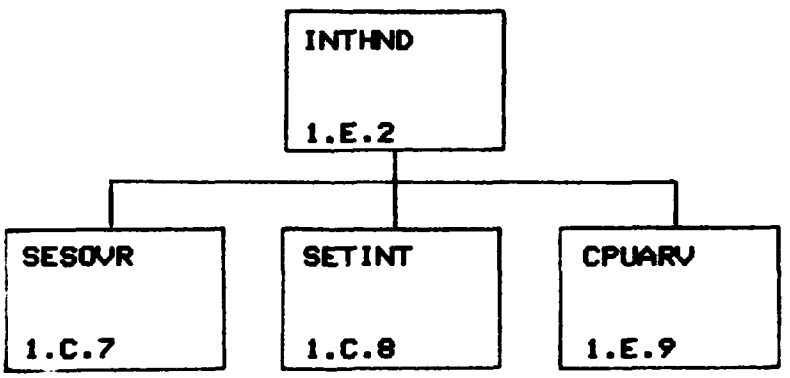


Figure F.3 INTHND Hierarchy.

end of run has been reached (ISTOP), EDPROC is called to set MSTOP to -1 which stops the simulation.

Event module INTHND processes all interactive sessions by routing jobs to the CPU and scheduling a user's think time (Figure F.3). If INTHND was scheduled by a CPU depar-

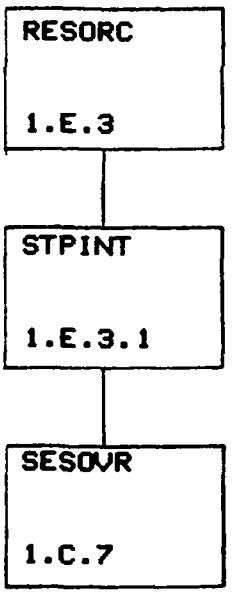


Figure F.4 RESORC Hierarchy.

ture (CPUDPT) and more jobs need to be run, SESOVR is called to conclude the session. If more jobs are left, a think time is calculated and INTHND is scheduled. At the end of a think time SETINT is called to set a job's attributes and CPUARV is called to schedule the jobs's completion. If the job was submitted in 'batch' mode and more jobs are left, another think time is calculated and INTHND is scheduled. If the job was submitted in batch mode and no jobs are left, the session is completed by calling SESOVR.

Figure F.4 illustrates RESORC hierarchy. RESORC is scheduled by CALSCH to 'open' and 'close' computers, printers, and card readers. The computers' operational status is set by changing the appropriate value in the XX array. If a computer is becoming not operational, STPINT is

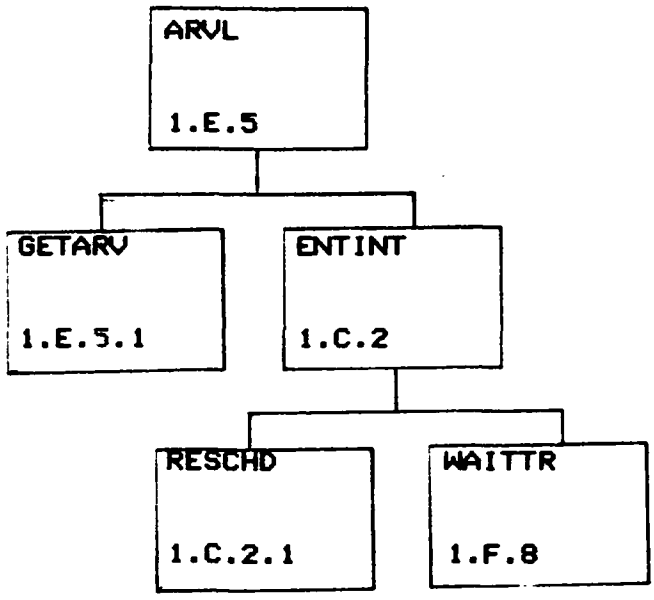


Figure F.5 ARVL Hierarchy.

called to search the event calendar for 'users' using the computer resource. Any users found are removed from the event calendar and routed to SESOVR. All printers and card readers attached to the computer are closed. Printer and card reader status changes are accomplished by calling the appropriate open or close SLAM subroutines.

Event module SETCPU is scheduled to change a CPU's states and set the session time whenever there is an interactive arrival. INTHND is scheduled.

All user arrivals to the network are routed to the network by event module ARVL as illustrated in Figure F.5. At each call the next arrival for the input location is scheduled using ARRATE. GETARV is called to set the arrival attributes such as computer, class, and so forth. If an interactive arrival, the event may be set to originate from

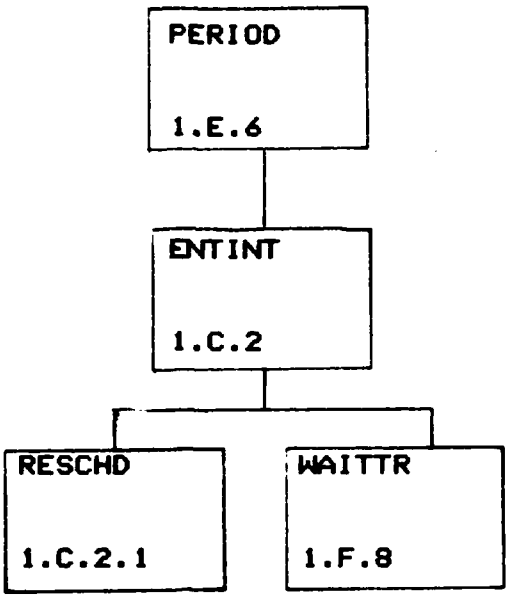


Figure F.6 PERIOD Hierarchy.

a dial-up location. After control is returned from GETARV, the event can be routed to either interactive (ENTINT) or batch. If batch, the event is entered into the network structure using the ENTER subroutine. If interactive, ENTINT is called to determine if the computer and terminal are available. If the computer is not available, the event is rescheduled (RESCHD). If the computer is available but a terminal is not available, WAITTR is called to determine if the user will wait. If the user will not wait, the user is rescheduled. Finally, if the computer and terminal are available or the user will wait, the arrival is entered into the network structure.

PERIOD module controls all of the periodic arrivals to the network as shown in Figure F.6. The module schedules

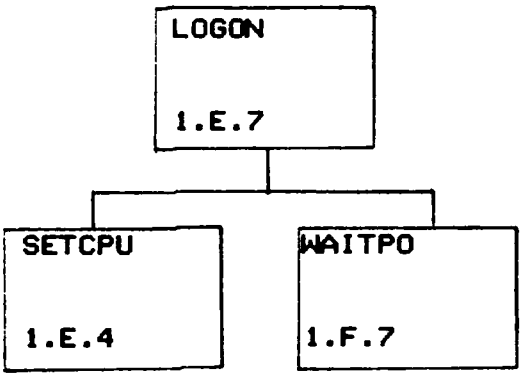


Figure F.7 LOGON Hierarchy.

the next periodic arrival by reading the next record in the CNSTxxy file. The current arrival is processed by calling ENTINT for interactive or subroutine ENTER for batch. ENTINT was described earlier.

Event module LOGON is scheduled by the network structure (Interactive Arrivals) when an arrival completes log on to a terminal (Figure F.7). If a port is available, SETCPU is called to update the CPU's state, calculate a session time, and schedule INTHND. Otherwise, WAITPO is called to determine if the user will wait for a port. If the user will not wait, the event is entered back into the network structure (Interactive Release) to release the terminal. Otherwise, the user is queued in the hold for port file.

Figure F.8 illustrates the event module CPUDPT hierarchy. CPUDPT handles all CPU departures and is scheduled by CPUARV. CPUDPT adjusts the CPU states; routes the job to a printer if word processing class; and starts a job in the hold queue. A job is routed to a printer by

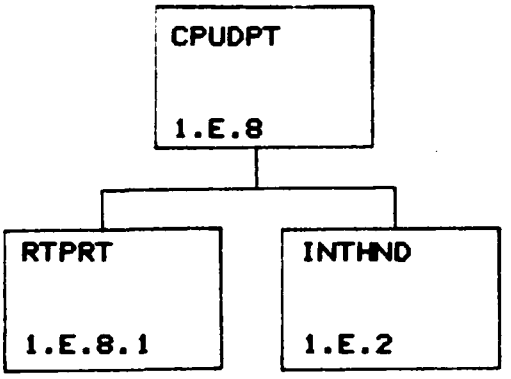


Figure F.8 CPUDPT Hierarchy.

calling RTPRT. RTPRT determines an appropriate printer and then enters the event into the network structure (Printers). If the job originated from an interactive location in wait mode, INTHND is called. Finally, if any jobs are in the CPU's hold queue, it is scheduled.

Event module CPUARV is scheduled for both interactive and batch jobs. For batch jobs, the network structure (Batch Arrivals) schedules the job directly. For interactive jobs, INTHND calls the module. In any case, the job's run parameters are calculated. If there is room in the 'execute' queue, CPUDPT is scheduled. Otherwise, the job is stored into the appropriate 'hold' queue.

All interactive arrivals that were rescheduled because a computer or terminal was not available are handled by event module LATARV (Figure F.9). Module ENTINT is called to process the arrivals as described earlier.

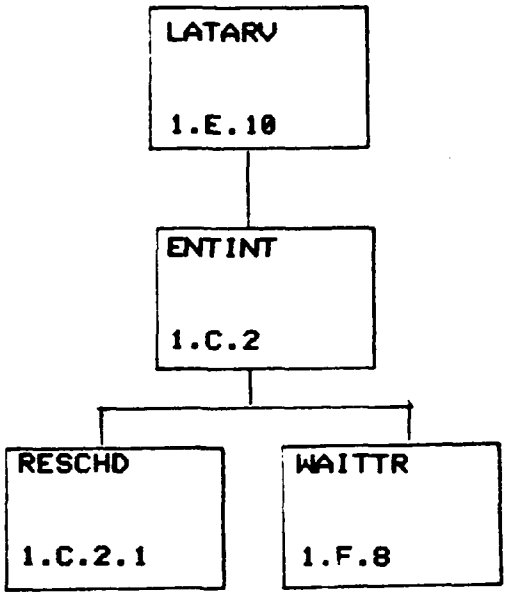


Figure F.9 LATARV Hierarchy.

Limitations

The network model's capacity is set by FORTRAN parameter statements. To increase the number of input locations, computers, classes, sizes, printers, card readers (gates), or wait probabilities, the appropriate parameter statement must be modified in each module it appears. The data dictionary provides a cross reference for each parameter. Additionally, modules UPDARV and UPDFRQ are structured to handle the output files created by the AFIT Workload Model. Any modifications to these files will require modifications to the UPDARV and UPDFRQ modules.

APPENDIX G

NETWORK MODEL DATA DICTIONARY

Name: ANYDST(MAXCLS,MAXCPU) Type: REAL
Common Block: ANYVAL
Description: Probabilities of using computer when 'any' specified.
Initialized by: RDPARM
Used by: GETARV,PRTCON,RDPARM

Name: ARRATE(MAXLOC,24) Type: REAL
Common Block: INVALS
Description: Current arrival rates for each location.
Initialized by: UPDARV
Used by: ARVL,INTLC,UPDARV

Name: BATINT(MAXCPU,MAXCLS) Type: REAL
Common Block: INTPRB
Description: Probabilities of submitting 'batch' jobs when in interactive.
Initialized by: RDPARM
Used by: PRTCON,RDPARM,SETINT

Name: CNVCPU(MAXCPU) Type: REAL
Common Block: CPSTAT
Description: Factor for variation of CPU seconds from computer to computer.
Initialized by: RDPARM
Used by: CPUARV,PRTCON,RDPARM

Name: CPULTM Type: REAL
Common Block: VARCOM
Description: Mean arrival time when computer is down.
Initialized by: BLOCK DATA/RDPARM
Used by: RDPARM,RESCHD

Name: CPULTS Type: REAL
Common Block: VARCOM
Description: Standard deviation arrival time when computer is down.
Initialized by: BLOCK DATA/RDPARM
Used by: RDPARM,RESCHD

Name: DIALUP(MAXCPU,24) Type: REAL
Common Block: INVALS
Description: Dial up probabilities for each computer over 24 hours.
Initialized by: RDPARM
Used by: PRTCON,RDPARM,GETARV

Name: DISTRB(MAXCLS,MAXSIZ,6,2) Type: REAL
Common Block: FACTOR
Description: Mean and standard deviation for the following factors:
1 - time till next job submittal
2 - number of K lines to print
3 - CPU seconds required
4 - Number of jobs this session
5 - memory size required
6 - I/O seconds required
Initialized by: RDPARM
Used by: PRTCON,RDPARM,SETCPU,CPUSEC,MEMSIZ,PRTLIN,XIOSEC

Name: DUPRT Type: REAL
Common Block: PRTPRM
Description: Probability of a print to dial up device.
Initialized by: RDPARM
Used by: PRTCON,RDPARM,RTPRT

Name: FRTABL(MAXLOC,MAXCPU,MAXCLS,MAXSIZ) Type: REAL
Common Block: INVALS
Description: Current cummulative frequency distribution for each input location.
Initialized by: UPDFRQ
Used by: UPDFRG,GETARV

Name: IADMN Type: INTEGER
Common Block: UNITS
Description: Faculty/Administration arrival and cummulative frequency distibution input file number.
Initialized by: BLOCK DATA
Used by: UPDARV,UPDFRQ

Name: IARVEV Type: INTEGER
Common Block: EVTCOD
Description: ARVL event code number.
Initialized by: BLOCK DATA
Used by: ARVL,INTLC

Name: IBATCH Type: INTEGER
Common Block: ENTCOD
Description: Batch arrival enter node number.
Initialized by: BLOCK DATA
Used by: ARVL,PERIOD

Name: IBATST Type: INTEGER
Common Block: OFFSET
Description: Start of input batch locations.
Initialized by: BLOCK DATA/RDPARM
Used by: ARVL,CPUARV,CPUDPT,PERIOD,RDPARM

Name: IBTCH Type: INTEGER
Common Block: UNITS
Description: Batch arrival and cummulative frequency
distribution input file number.
Initialized by: BLOCK DATA
Used by: INTLC,UPDARV,UPDFRQ

Name: ICLKEV Type: INTEGER
Common Block: EVTCOD
Description: CLOCK event code number.
Initialized by: BLOCK DATA
Used by: CLOCK,INTLC

Name: ICONST Type: INTEGER
Common Block: UNITS
Description: Constant input file number.
Initialized by: BLOCK DATA
Used by: INTLC,PERIOD

Name: ICPDEV Type: INTEGER
Common Block: EVTCOD
Description: CPUDPT event code number.
Initialized by: BLOCK DATA
Used by: CPUARV,CPUDPT

Name: ICPU Type: INTEGER
Common Block: CPSTAT
Description: Offset into computer states for CPU seconds.
Initialized by: BLOCK DATA
Used by: CPUARV,CPUDPT,FIGUTL

Name: ICPUOS Type: INTEGER
Common Block: OFFSET
Description: Offset into XX array for CPU up/down flags.
Initialized by: BLOCK DATA
Used by: CALSCH,ENTINT,RDPARM,RESORC,RTPRT

Name: ICRGAT(MAXCPU,MAXGAT) Type: INTEGER
Common Block: SCHEDL
Description: Identifies for every computer which card reader
gates open/close as computers come up/down.
Initialized by: BLOCK DATA/RDPARM
Used by: PRTCON,RDPARM,RESORC

Name: IEXJOB Type: INTEGER
Common Block: CPSTAT
Description: Offset into computer states for number of
executing jobs.
Initialized by: BLOCK DATA
Used by: CPUARV,CPUDPT

Name: IFACT Type: INTEGER
Common Block: UNITS
Description: Input FACTOR file unit number.
Initialized by: BLOCK DATA
Used by: RDPARM

Name: IGATOS Type: INTEGER
Common Block: OFFSET
Description: Offset into start of gate numbers.
Initialized by: BLOCK DATA
Used by: CALSCH,RESORC

Name: IHLDOS Type: INTEGER
Common Block: OFFSET
Description: Offset into files for CPU hold queue.
Initialized by: BLOCK DATA
Used by: CPUARV,CPUDPT

Name: IMEM Type: INTEGER
Common Block: CPSTAT
Description: Offset into computer states for current memory
requirements.
Initialized by: BLOCK DATA
Used by: CPUARV,CPUDPT

Name: IMPL Type: INTGER
Common Block: CPSTAT
Description: Offset into computer states for multiprogram
level.
Initialized by: BLOCK DATA
Used by: CPUARV,CPUDPT,RDPARM

Name: INOUT Type: INTEGER
Common Block: CPSTAT
Description: Offset into computer states for total I/O
seconds.
Initialized by: BLOCK DATA
Used by: CPUARV,CPUDPT,FIGUTL

Name: INTARV Type: INTEGER
Common Block: ENTCOD
Description: Interactive arrival enter node number.
Initialized by: BLOCK DATA
Used by: ENTINT

Name: INTHEV Type: INTEGER
Common Block: EVTCOD
Description: INTHND event code number.
Initialized by: BLOCK DATA
Used by: INTHND,RTPRT,SETCPU,STPINT

Name: IOTHR Type: INTEGER
Common Block: UNITS
Description: Other arrival and cummulative frequency
distribution input file unit number.
Initialized by: BLOCK DATA
Used by: UPDARV,UPDFRQ

Name: IPEREV Type: INTEGER
Common Block: EVTCOD
Description: PERIOD event code number.
Initialized by: BLOCK DATA
Used by: INTLC,PERIOD

Name: IPORTS Type: INTEGER
Common Block: OFFSET
Description: Offset into XX array for start of computer
port counters.
Initialized by: BLOCK DATA
Used by: LOGON,PRTCON,RDPARM,SESOVR,SETCPU,WAITPO

Name: IPRGAT(MAXGAT,MAXPRT) Type: INTEGER
Common Block: SCHEDL
Description: Indentifies the printers which open/close as
the gate locations open/close.
Initialized by: BLOCK DATA/RDPARM
Used by: PRTCON,RDPARM,RESORC

Name: IPRTEN Type: INTEGER
Common Block: ENTCOD
Description: Printers enter node number.
Initialized by: BLOCK DATA
Used by: RTPRT

Name: IPRTOS Type: INTEGER
Common Block: OFFSET
Description: Offset to start of printer resource numbers.
Initialized by: BLOCK DATA
Used by: RESORC

Name: IRESEV Type: INTEGER
Common Block: EVTCOD
Description: RESORC event code number.
Initialized by: BLOCK DATA
Used by: CALSCH

Name: IRTSTR Type: INTEGER
Common Block: OFFSET
Description: Start of responce time collect numbers.
Initialized by: BLOCK DATA
Used by: CPUDPT

Name: ISTCEV Type: INTEGER
Common Block: EVTCOD
Description: SETCPU event code number.
Initialized by: BLOCK DATA
Used by: SESOVR

Name: ISTUD Type: INTEGER
Common Block: UNITS
Description: Student arrival rates and frequency
distributions input file unit number.
Initialized by: BLOCK DATA
Used by: INTLC,UPDARV,UPDFRQ

Name: ITRMAC Type: INTEGER
Common Block: CPSTAT
Description: Offset into computer states for number of
terminals active.
Initialized by: BLOCK DATA
Used by: CPUARV,INTHND,SESOVR,SETCPU

Name: ITRMOL Type: INTEGER
Common Block: CPSTAT
Description: Offset into computer states for number of terminals online.
Initialized by: BLOCK DATA
Used by: SESOVR,SETCPU

Name: ITRMOS Type: INTEGER
Common Block: OFFSET
Description: Offset to start of terminal resource number.
Initialized by: BLOCK DATA
Used by: ENTINT,WAITTR

Name: ITRMWP Type: INTEGER
Common Block: CPSTAT
Description: Offset into computer states for number of terminals word processing.
Initialized by: BLOCK DATA
Used by: SESOVR,SETCPU

Name: LATEVT Type: INTEGER
Common Block: EVTCOD
Description: LATARV event code number.
Initialized by: BLOCK DATA
Used by: RESCHD,WAITPO,WAITTR

Name: LOGFLG Type: LOGICAL
Common Block: DIMEN
Description: Log Flag. 0/Off 1/On
Initialized by: RDPARM
Used by: ARVL,CPUARV,CPUDPT,DYPROC,ENTINT,HRPROC,INTHND, LATARV,LOGON,PERIOD,RDPARM,RESCHD,RESORC,RTPRT, SESOVR,SETCPU,SETINT,STPINT,UPDARV,WAITPO,WAITTR

Name: LOGCNT(25) Type: INTEGER
Common Block: LOGPRM
Description: Log counter.
Initialized by: BLOCK DATA
Used by: HRPROC,LOG

Name: LOGON(25) Type: INTEGER
Common Block: LOGPRM
Description: Individual log on/off flag. 0/off, 1/on
Initialized by: BLOCK DATA/RDPARM
Used by: LOG,RDPARM

Name: MAXCLS Type: PARAMETER
Description: Maximum number of work classes.
Used by: ARVL,BLOCK DATA,GETARV,INTLC,PRTCON,RDPARM,SETCPU,
SETINT,UPDARV,UPDFRQ,CPUSEC,MEMSIZ,PRTLIN,XIOSEC

Name: MAXCPU Type: PARAMETER
Description: Maximum number of computers.
Used by: ARVL,BLOCK DATA,CALSCH,CPUDPT,FIGUTL,GETARV,
INTHND,INTLC,PRTCON,RDPARM,RESORC,RTPRT,SESOVR,
SETCPU,SETINT,UPDARV,UPDFRQ

Name: MAXGAT Type: PARAMETER
Description: Maximum number of batch gates.
Used by: BLOCK DATA,CLASCH,PRTCON,RDPARM,RESORC

Name: MAXLOC Type: PARAMETER
Description: Maximum number of arrival locations.
Used by: ARVL,BLOCK DATA,GETARV,INTLC,PRTCON,RDPARM,RTPRT,
UPDARV,UPDFRQ,WAITPO,WAITTR

Name: MAXPRT Type: PARAMETER
Description: Maximum number of printers.
Used by: BLOCK DATA,CALSCH,PRTCON,RDPARM,RESORC,RTPRT

Name: MAXSIZ Type: PARAMETER
Description: Maximum number of work sizes.
Used by: ARVL,BLOCK DATA,GETARV,INTLC,PRTCON,RDPARM,SETCPU,
UPDARV,UPDFRQ,CPUSEC,MEMSIZ,PRTLIN,XIOSEC

Name: MAXWAT Type: PARAMETER
Description: Maximum number of work classes.
Used by: BLOCK DATA,RDPARM,WAITPO,WAITTR

Name: NRLSIN Type: INTEGER
Common Block: ENTCOO
Description: Release interactive enter node number.
Initialized by: BLOCK DATA
Used by: LOGON,SESOVR

Name: NUMCLS Type: INTEGER
Common Block: DIMEN
Description: Number of work classes.
Initialized by: BLOCK DATA/RDPARM
Used by: GETARV,INTLC,PRTCON,RDPARM

Name: NUMCPU Type: INTEGER
Common Block: DIMEN
Description: Number of computers.
Initialized by: BLOCK DATA/RDPARM
Used by: CALSCH,FIGUTL,GETARV,INTLC,PRTCON,RDPARM

Name: NUMGAT Type: INTEGER
Common Block: DIMEN
Description: Number of batch gates.
Initialized by: BLOCK DATA/RDPARM
Used by: CALSCH,PRTCON,RDPARM,RESORC

Name: NUMLOC Type: INTEGER
Common Block: DIMEN
Description: Number of arrival locations.
Initialized by: BLOCK DATA/RDPARM
Used by: INTLC,PRTCON,RDPARM

Name: NUMPRT Type: INTEGER
Common Block: DIMEN
Description: Number of printers.
Initialized by: BLOCK DATA/RDPARM
Used by: PRTCON,RDPARM,RESORC,RTPRT

Name: NUMSIZ Type: INTEGER
Common Block: DIMEN
Description: Number of work sizes.
Initialized by: BLOCK DATA/RDPARM
Used by: GETARV,INTLC,PRTCON,RDPARM

Name: ONEJOB Type: REAL
Common Block: INTPRB
Description: Probability of printing when only one job left for interactive session.
Initialized by: RDPARM
Used by: PRTCON,RDPARM,SETINT

Name: OPRNM Type: REAL
Common Block: VARCOM
Description: Mean of batch operator input time.
Initialized by: BLOCK DATA/RDPARM
Used by: RDPARM,USERF

Name: OPRSTD Type: REAL
Common Block: VARCOM
Description: Standard deviation of batch operator input time.
Initialized by: BLOCK DATA/RDPARM
Used by: RDPARM,USERF

Name: PCPRTM Type: REAL
Common Block: PRTPRM
Description: Dial-up mean print speed.
Initialized by: BLOCK DATA/RDPARM
Used by: RDPARM,RTPRT

Name: PCPRTS Type: REAL
Common Block: PRTPRM
Description: Dial-up standard deviation print speed.
Initialized by: BLOCK DATA/RDPARM
Used by: RDPARM,RTPRT

Name: PORTWT(MAXWAT) Type: REAL
Common Block: WAITCM
Description: Probabilities of waiting for a computer port.
Initialized by: BLOCK DATA/RDPARM
Used by: RDPARM,WAITPO

Name: PRTSPD(MAXPRT) Type: REAL
Common Block: PRTPRM
Description: Printer speeds (lines/minute).
Initialized by: RDPARM
Used by: PRTCON,RDPARM,RTPRT

Name: PRTTAB(MAXCPU,MAXLOC,MAXPRT) Type: REAL
Common Block: PRTPRM
Description: Table of probabilities of selection of printer
by computer and location.
Initialized by: RDPARM
Used by: PRTCON,RDPARM,RTPRT

Name: SCHBAT(MAXGAT,7,2) Type: REAL
Common Block: SCHEDL
Description: Batch locations weekly schedule.
Initialized by: BLOCK DATA/RDPARM
Used by: CALSCH,PRTCON,RDPARM

Name: SCHCPU(MAXCPU,7,2) Type: REAL
Common Block: SCHEDL
Description: Computer's weekly schedule.
Initialized by: BLOCK DATA/RDPARM
Used by: CALSCH,PRTCON,RDPARM

Name: TRMLTM Type: REAL
Common Block: VARCOM
Description: Mean arrival time when a terminal is not
available.
Initialized by: BLOCK DATA/RDPARM
Used by: RDPARM,WAITTR

Name: TRMLTS Type: REAL
Common Block: VARCOM
Descriptin: Standard deviation or the arrival time when a terminal is not available.
Initialized by: BLOCK DATA/RDPARM
Used by: RDPARM,WAITTR

Name: TRMWAT(MAXLOC,MAXWAT) Type: REAL
Common Block: VARCOM
Description: Probabilities of waiting for a terminal for each location.
Initialized by: BLOCK DATA/RDPARM
Used by: RDPARM,WAITTR

Name: WAITMN Type: REAL
Common Block: WAITCM
Description: Mean time to reschedule arrival waiting for port.
Initialized by: BLOCK DATA/RDPARM
Used by: RDRARM,WAITPO

Name: WAITST Type: REAL
Common Block: WAITCM
Description: Standard deviation of time to reschedule arrival waiting for port.
Initialized by: BLOCK DATA/RDPARM
Used by: RDPARM,WAITPO

Name: WPPROB(MAXCLS) Type: REAL
Common Block: INTPRB
Description: Probabilities of changing class to word processing class (print).
Initialized by: RDPARM
Used by: PRTCON,RDPARM,SETINT

Name: XLOGMN Type: REAL
Common Block: VARCOM
Description: Mean log on time.
Initialized by: BLOCK DATA/RDPARM
Used by: RDPARM,USERF

Name: XLOGST Type: REAL
Common Block: VARCOM
Description: Standard Deviation of the log on time.
Initialized by: BLOCK DATA/RDPARM
Used by: RDPARM,USERF

APPENDIX H

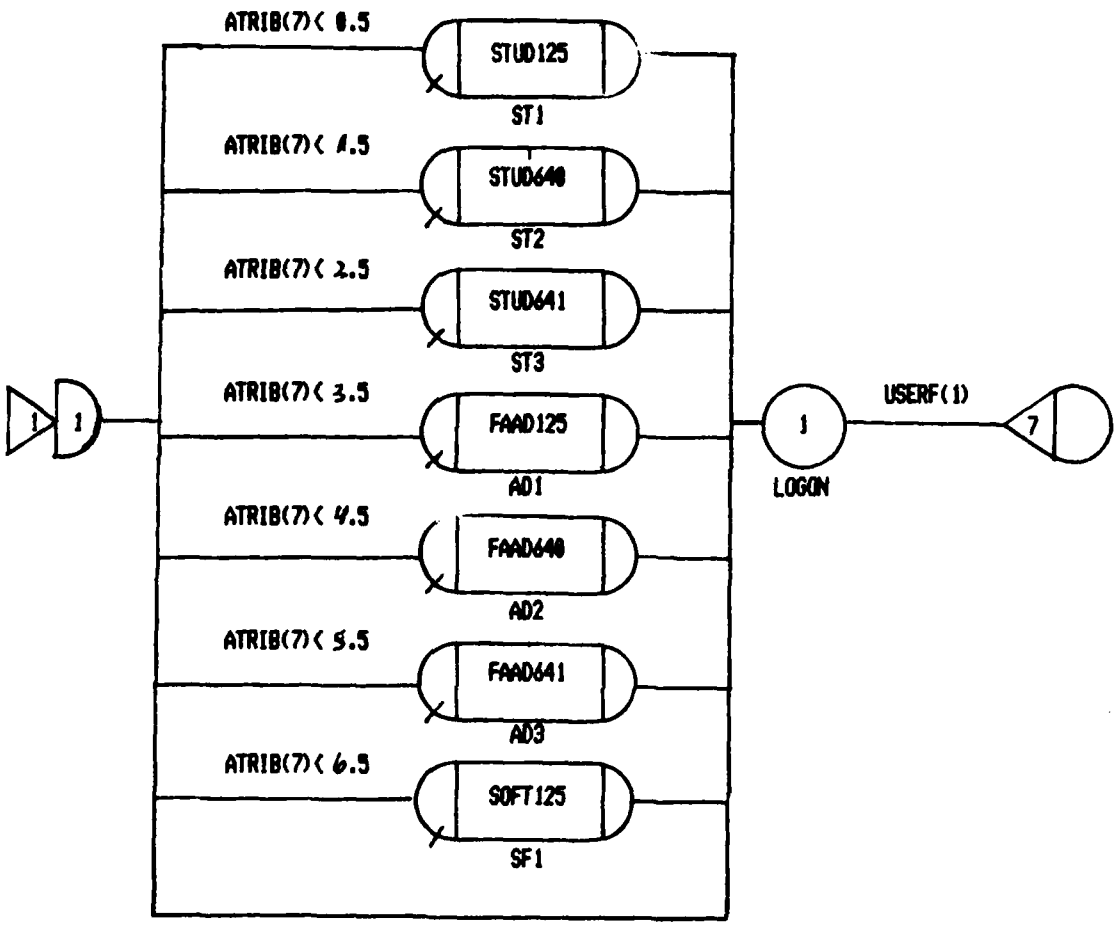
NETWORK MODEL 'NETWORK' DIAGRAMS AND SOURCE CODE LISTING

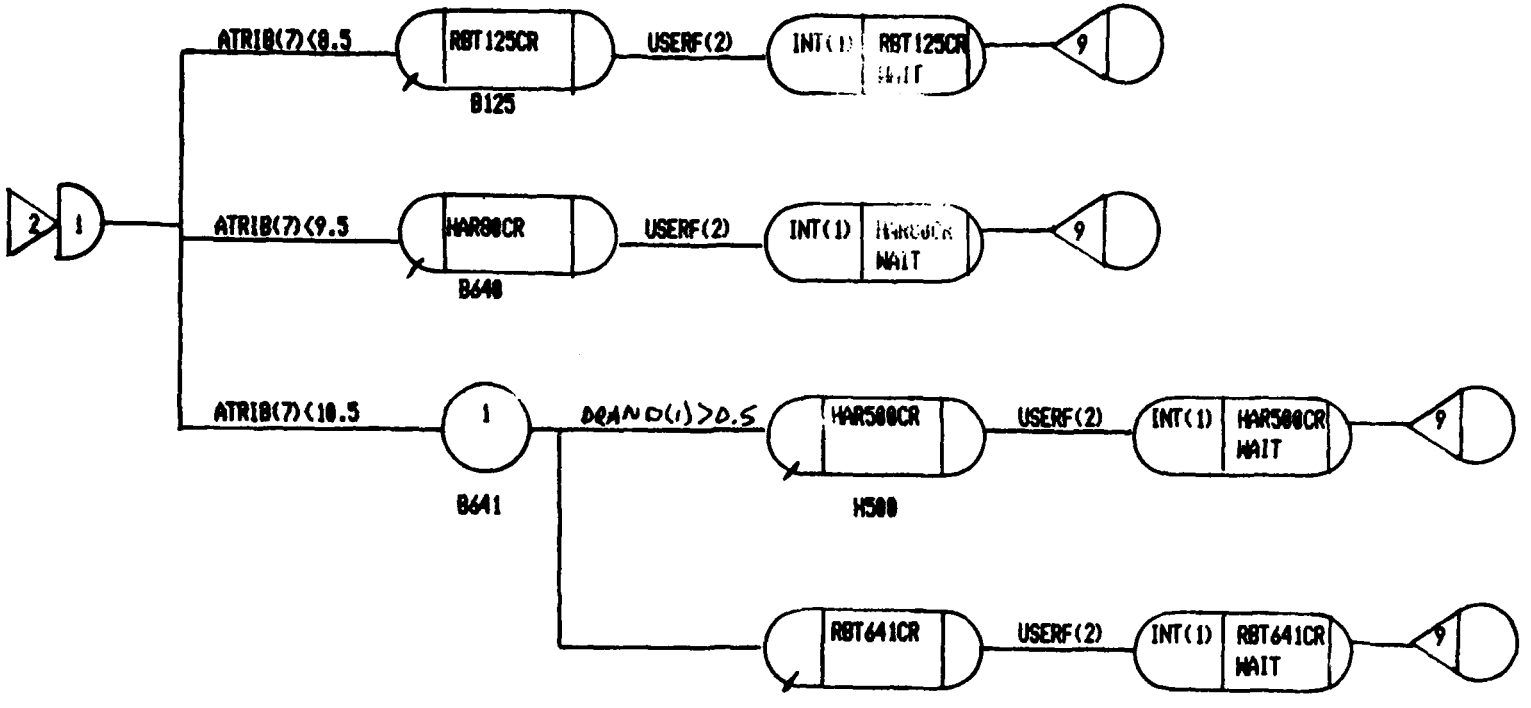
RESOURCE BLOCKS

STUD125(20)	10
STUD640(20)	11
STUD641(20)	12
FAAD125(20)	13
FAAD640(20)	14
FAAD641(20)	15
SOFT125(20)	16
RBT125LP(1)	20
RBT641LP(1)	21
HAR80LP(1)	22
HAR80PEN(1)	23
HAR80PP(1)	24
SSCPP(1)	25
HAR500LP(1)	26

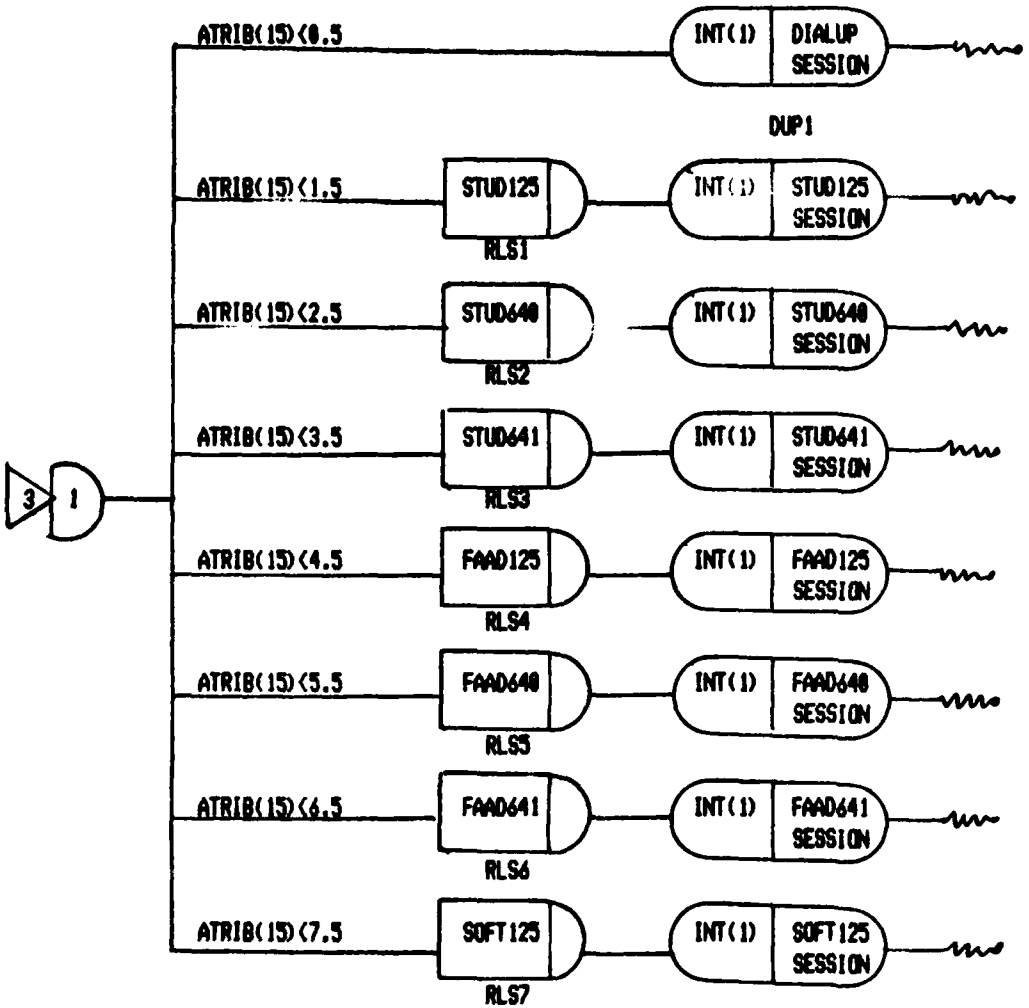
GATE BLOCKS

RBT125CR	CLOSE	30
HAR80CR	CLOSE	31
RBT125CR	OPEN	32
HAR500CR	CLOSE	33

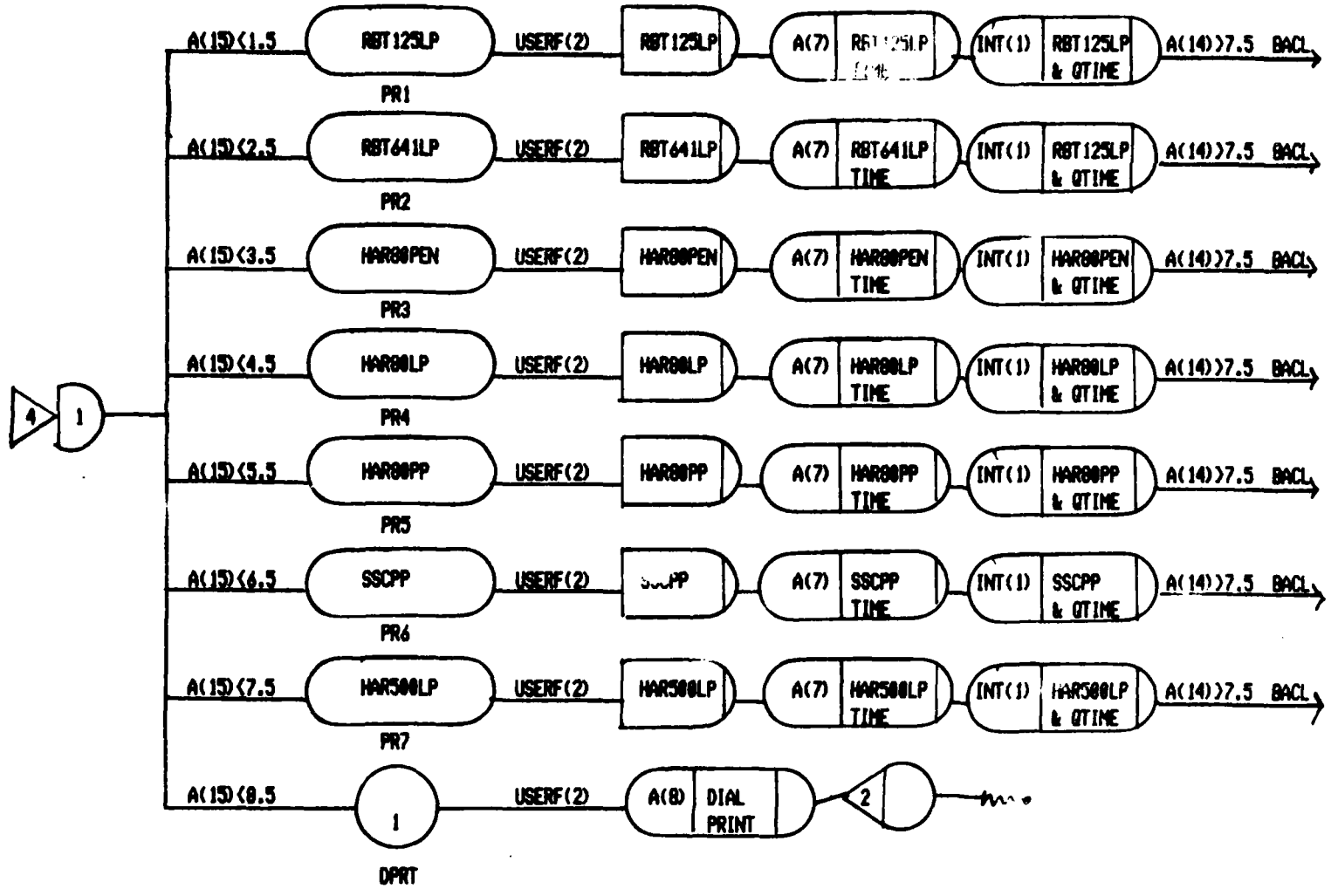




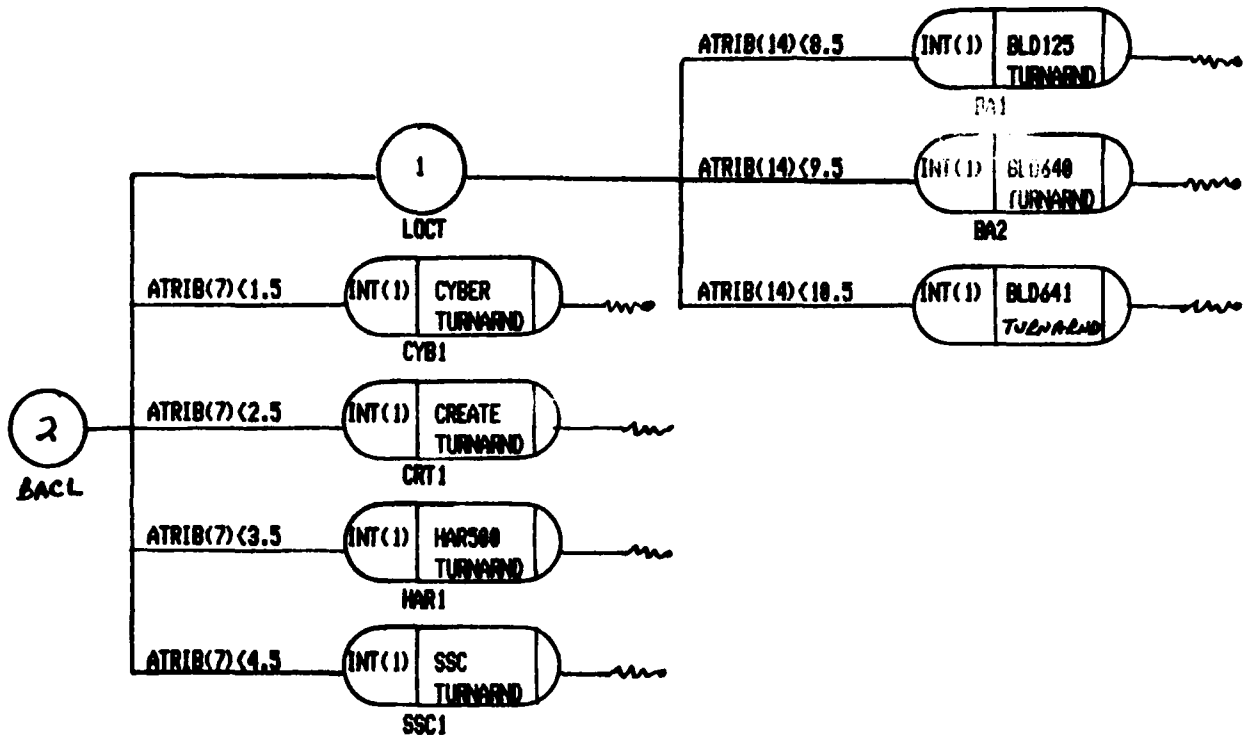
Batch Arrivals



Interactive Release


A(15)<1.5
RBT125LP
USERF(2)
RBT125LP
A(7)
RBT125LP
TIME
INT(1)
RBT125LP
& QTIME
A(14)>7.5 BACL
PR1
A(15)<2.5
RBT641LP
USERF(2)
RBT641LP
A(7)
RBT641LP
TIME
INT(1)
RBT125LP
& QTIME
A(14)>7.5 BACL
PR2
A(15)<3.5
HAR88PEN
USERF(2)
HAR88PEN
A(7)
HAR88PEN
TIME
INT(1)
HAR88PEN
& QTIME
A(14)>7.5 BACL
PR3
A(15)<4.5
HAR88LP
USERF(2)
HAR88LP
A(7)
HAR88LP
TIME
INT(1)
HAR88LP
& QTIME
A(14)>7.5 BACL
PR4
A(15)<5.5
HAR88PP
USERF(2)
HAR88PP
A(7)
HAR88PP
TIME
INT(1)
HAR88PP
& QTIME
A(14)>7.5 BACL
PR5
A(15)<6.5
SSCPP
USERF(2)
A(7)
SSCPP
TIME
INT(1)
SSCPP
& QTIME
A(14)>7.5 BACL
PR6
A(15)<7.5
HAR500LP
USERF(2)
HAR500LP
A(7)
HAR500LP
TIME
INT(1)
HAR500LP
& QTIME
A(14)>7.5 BACL
PR7
A(15)<8.5
1
USERF(2)
A(8)
DIAL
PRINT
2
DPRT
4
1

Printers



Collect Batch Statistics

```
GEN,MCDERMOTT,NETWORK,09/20/83,1,Y,Y,Y,N;
LIMITS,44,15,500;
TIMST,XX(10),CYBER AVAIL;
TIMST,XX(11),CREATE AVAIL;
TIMST,XX(12),HAR500 AVAIL;
TIMST,XX(13),SSC AVAIL;
TIMST,XX(40),CYBER PORTS;
TIMST,XX(41),CREATE PORTS;
TIMST,XX(42),HAR500 PORTS;
STAT,1,CYBER RESPONSE;
STAT,2,CREATE RESPONSE;
STAT,3,HAR500 RESPONSE;
STAT,4,SSC RESPONSE;
INIT,0.,6000000.;
;
NETWORK;
;
;  RESOURCES AND GATES
;       (FILES 1-10 RESERVED FOR DISCRETE USE)
;
;       *** TERMINALS ***
;
      RESOURCE/STUD125(20),10;
      RESOURCE/STUD640(20),11;
      RESOURCE/STUD641(20),12;
      RESOURCE/FAAD125(20),13;
      RESOURCE/FAAD640(20),14;
      RESOURCE/FAAD641(20),15;
      RESOURCE/SOFT125(20),16;
;
;       *** PRINTERS ***
;
      RESOURCE/RBT125LP(0),20;
      RESOURCE/RBT641LP(1),21;
      RESOURCE/HAR80PEN(0),22;
      RESOURCE/HAR80LP(0),23;
      RESOURCE/HAR80PP(0),24;
      RESOURCE/SSCPP(1),25;
      RESOURCE/HAR500LP(0),26;
;
;       *** BATCH LOCATIONS ***
;
      GATE/RBT125CR,CLOSE,30;
      GATE/HAR80CR,CLOSE,31;
      GATE/RBT641CR,OPEN,32;
      GATE/HAR500CR,CLOSE,33;
;
; BEGIN NETWORK
;
;
;  **** INTERACTIVE ARRIVALS ****
;
      ENTER,1,1;
      ACT,,ATRIB(7).LT.0.5,LOGON;          DIALUP
      ACT,,ATRIB(7).LT.1.5,ST1;           STUD TERM 125
```

```
      ACT,,ATRIB(7).LT.2.5,ST2;          STUD TERM 640
      ACT,,ATRIB(7).LT.3.5,ST3;          STUD TERM 641
      ACT,,ATRIB(7).LT.4.5,AD1;          FAC/ADMIN 125
      ACT,,ATRIB(7).LT.5.5,AD2;          FAC/ADMIN 640
      ACT,,ATRIB(7).LT.6.5,AD3;          FAC/ADMIN 641
      ACT,,ATRIB(7).LT.7.5,SF1;          SOFT DEV 125
      TERM;
;
ST1   AWAIT,STUD125,1;                   GET A TERMINAL
      ACT,,,LOGON;
;
ST2   AWAIT,STUD640,1;                   GET A TERMINAL
      ACT,,,LOGON;
;
ST3   AWAIT,STUD641,1;                   GET A TERMINAL
      ACT,,,LOGON;
;
AD1   AWAIT,FAAD125,1;                   GET A TERMINAL
      ACT,,,LOGON;
;
AD2   AWAIT,FAAD640,1;                   GET A TERMINAL
      ACT,,,LOGON;
;
AD3   AWAIT,FAAD641,1;                   GET A TERMINAL
      ACT,,,LOGON;
;
SF1   AWAIT,SOFT125.1;                   GET A TERMINAL
      ACT,,,LOGON;
;
; LOGON
;
LOGON GOON;
      ACT,USERF(1);                      LOG ON TIME
      EVENT,7;                           LOG ON COMPLETE
      TERM;
;
;      *** BATCH ARRIVALS ***
;
      ENTER,2,1;
      ACT,,ATRIB(7).LT.8.5,B125;          BATCH LOCATION BLDG 125
      ACT,,ATRIB(7).LT.9.5,B640;          BATCH LOCATION BLDG 640
      ACT,,ATRIB(7).LT.10.5,B641;         BATCH LOCATION BLDG 641
      TERM;
;
B125  AWAIT,RBT125CR;                    CARD READER AVAILABLE?
      ACT,USERF(2);                      WAIT FOR OPERATOR INPUT
      COLCT,INT(1),CR125 WAIT;           COLLECT WAIT TIME
      EVENT,9;                           INPUT TO CPU
      TERM;
;
B640  AWAIT,HAR80CR;                     CARD READER AVAILABLE?
      ACT,USERF(2);                      WAIT FOR OPERATOR INPUT
      COLCT,INT(1),HAR80CR WAIT;         COLLECT WAIT TIME
      EVENT,9;                           INPUT TO CPU
      TERM;
```

```
;
B641  GOON,1;
      ACT,,ATRIB(3).GT.2.5.AND.DRAND(1).GT.0.5,H500;
;                                          RBT641 OR HAR500 BATCH?
      ACT;
      AWAIT,RBT641CR;                      CARD READER AVAILABLE?
      ACT,USERF(2);                        WAIT FOR OPERATOR INPUT
      COLCT,INT(1),CR641 WAIT;             COLLECT WAIT TIME
      EVENT,9;                             INPUT TO CPU
      TERM;
;
H500  AWAIT,HAR500CR;                      CARD READER AVAILABLE?
      ACT,USERF(2);                        WAIT FOR OPERATOR INPUT
      COLCT,INT(1),HAR500CR WAIT;          COLLECT WAIT TIME
      EVENT,9;                             INPUT TO CPU
      TERM;
;
;  INTERACTIVE RELEASE
;
      ENTER,3,1;
      ACT,,ATRIB(7).LT.0.5,DUP1;           DIALUP
      ACT,,ATRIB(7).LT.1.5,RLS1;            STUDENT BLDG 125
      ACT,,ATRIB(7).LT.2.5,RLS2;            STUDENT BLDG 640
      ACT,,ATRIB(7).LT.3.5,RLS3;            STUDENT BLDG 641
      ACT,,ATRIB(7).LT.4.5,RLS4;            FAC/ADMIN BLDG 125
      ACT,,ATRIB(7).LT.5.5,RLS5;            FAC/ADMIN BLDG 640
      ACT,,ATRIB(7).LT.6.5,RLS6;            FAC/ADMIN BLDG 641
      ACT,,ATRIB(7).LT.7.5,RLS7;            SOFT DEV BLDG 125
      TERM;
;
DUP1  COLCT,INT(1),DIALUP SESSION;
      TERM;
RLS1  FREE,STUD125;                        FREE TERMINAL
      COLCT,INT(1),STUD125 SESSION;        COLLECT SESSION TIME
      TERM;
RLS2  FREE,STUD640;                        FREE TERMINAL
      COLCT,INT(1),STUD640 SESSION;        COLLECT SESSION TIME
      TERM;
RLS3  FREE,STUD641;                        FREE TERMINAL
      COLCT,INT(1),STUD641 SESSION;        COLLECT SESSION TIME
      TERM;
RLS4  FREE,FAAD125;                        FREE TERMINAL
      COLCT,INT(1),FAAD125 SESSION;        COLLECT SESSION TIME
      TERM;
RLS5  FREE,FAAD640;                        FREE TERMINAL
      COLCT,INT(1),FAAD640 SESSION;        COLLECT SESSION TIME
      TERM;
RLS6  FREE,FAAD641;                        FREE TERMINAL
      COLCT,INT(1),FAAD641 SESSION;        COLLECT SESSION TIME
      TERM;
RLS7  FREE,SOFT125;                        FREE TERMINAL
      COLCT,INT(1),SOFT125 SESSION;        COLLECT SESSION TIME
      TERM;
;
;  PRINTERS
```

```
;
      ENTER,4,1;
      ACT,,ATRIB(15).LE.1.5,PR1;
      ACT,,ATRIB(15).LE.2.5,PR2;
      ACT,,ATRIB(15).LE.3.5,PR3;
      ACT,,ATRIB(15).LE.4.5,PR4;
      ACT,,ATRIB(15).LE.5.5,PR5;
      ACT,,ATRIB(15).LE.6.5,PR6;
      ACT,,ATRIB(15).LE.7.5,PR7;
      ACT,,ATRIB(15).LE.8.5,DPRT;
;
;      **** RBT125 LINE PRINTER ****
;
PR1   AWAIT,RBT125LP;                    WAIT FOR PRINTER
      ACT,ATRIB(8);                      TIME FUNCTION OF PRINTER
      FREE,RBT125LP;                     FREE PRINTER
      COLCT,ATRIB(8),RBT125LP TIME;      COLLECT PRINTER TIME
      COLCT,INT(6),RBT125LP & QTIME,,1;  COLLECT PRINTER + QUEUE TIME
      ACT,,ATRIB(14).GT.7.5,BACL;        WAS THIS BATCH INPUT?
      TERM;
;
;      **** RBT641 LINE PRINTER ****
;
PR2   AWAIT,RBT641LP;                    WAIT FOR PRINTER
      ACT,ATRIB(8);                      TIME FUNCTION OF PRINTER
      FREE,RBT641LP;                     FREE PRINTER
      COLCT,ATRIB(8),RBT641LP TIME;      COLLECT PRINTER TIME
      COLCT,INT(6),RBT641LP & QTIME,,1;  COLLECT PRINTER + QUEUE TIME
      ACT,,ATRIB(14).GT.7.5,BACL;        WAS THIS BATCH INPUT?
      TERM;
;
;      **** HARRIS 80 PEN/PLOTTER ****
;
PR3   AWAIT,HAR80PEN;                    WAIT FOR PRINTER
      ACT,ATRIB(8);                      TIME FUNCTION OF PRINTER
      FREE,HAR80PEN;                     FREE PRINTER
      COLCT,ATRIB(8),HAR80PEN TIME;      COLLECT PRINTER TIME
      COLCT,INT(6),HAR80PEN & QTIME,,1;  COLLECT PRINTER + QUEUE TIME
      ACT,,ATRIB(14).GT.7.5,BACL;        WAS THIS BATCH INPUT?
      TERM;
;
;      **** HARRIS 80 LINE PRINTER ****
;
PR4   AWAIT,HAR80LP;                     WAIT FOR PRINTER
      ACT,ATRIB(8);                      TIME FUNCTION OF PRINTER
      FREE,HAR80LP;                      FREE PRINTER
      COLCT,ATRIB(8),HAR80LP TIME;       COLLECT PRINTER TIME
      COLCT,INT(6),HAR80LP & QTIME,,1;   COLLECT PRINTER + QUEUE TIME
      ACT,,ATRIB(14).GT.7.5,BACL;        WAS THIS BATCH INPUT?
      TERM;
;
;      **** HARRIS 80 PRINTER PLOTTER ****
;
PR5   AWAIT,HAR80PP;                     WAIT FOR PRINTER
      ACT,ATRIB(8);                      TIME FUNCTION OF PRINTER
```

```
        FREE,HAR80PP;                       FREE PRINTER
        COLCT,ATRIB(8),HAR80PP TIME;        COLLECT PRINTER TIME
        COLCT,INT(6),HAR80PP & QTIME,,1;      COLLECT PRINTER + QUEUE TIME
        ACT,,ATRIB(14).GT.7.5,BACL;           WAS THIS BATCH INPUT?
        TERM;
;
;       **** SSC PRINTER PLOTTER ****
;
PR6     AWAIT,SSCPP;                        WAIT FOR PRINTER
        ACT,ATRIB(8);                       TIME FUNCTION OF PRINTER
        FREE,SSCPP;                         FREE PRINTER
        COLCT,ATRIB(8),SSCPP TIME;          COLLECT PRINTER TIME
        COLCT,INT(6),SSCPP & QTIME,,1;        COLLECT PRINTER + QUEUE TIME
        ACT,,ATRIB(14).GT.7.5,BACL;           WAS THIS BATCH INPUT?
        TERM;
;
;       **** HARRIS 500 LINE PRINTER ****
;
PR7     AWAIT,HAR500LP;                     WAIT FOR PRINTER
        ACT,ATRIB(8);                       TIME FUNCTION OF PRINTER
        FREE,HAR500LP;                      FREE PRINTER
        COLCT,ATRIB(8),HAR500LP TIME;       COLLECT PRINTER TIME
        COLCT,INT(6),HAR500LP & QTIME,,1;     COLLECT PRINTER + QUEUE TIME
        ACT,,ATRIB(14).GT.7.5,BACL;           WAS THIS BATCH INPUT?
        TERM;
;
; DIALUP PRINTER
;
DPRT    GOON,1;
        ACT,ATRIB(8);
        COLCT,ATRIB(8),DIAL PRINT;
        EVENT,2;
        TERM;
;
;       **** COLLECT BATCH STATISTICS ****
;
BACL    GOON,2;
        ACT,,,LOCT;
;       COLLECT COMPUTER TURNAROUND TIMES
        ACT,,ATRIB(7).LT.1.5,CYB1;           CYBER
        ACT,,ATRIB(7).LT.2.5,CRE1;          CREATE
        ACT,,ATRIB(7).LT.3.5,HAR1;          HARRIS 500
        ACT,,ATRIB(7).LT.4.5,SSC1;          SSC
;       COLLECT LOCATION TURNAROUND TIMES
LOCT    GOON,1;
        ACT,,ATRIB(14).LT.8.5,BA1;           BUILDING 125
        ACT,,ATRIB(14).LT.9.5,BA2;           BUILDING 640
        ACT,,ATRIB(14).LT.10.5,BA3;          BUILDING 641
        TERM;
;
CYB1    COLCT,INT(1),CYBER TURNARND;
        TERM;
CRE1    COLCT,INT(1),CREATE TURNARND;
        TERM;
HAR1    COLCT,INT(1),HAR500 TURNARND;
```

```
      TERM;
SSC1  COLCT,INT(1),SSC TURNAROND;
      TERM;
;
BA1   COLCT,INT(1),BLD125 TURNAROND;
      TERM;
BA2   COLCT,INT(1),BLD640 TURNAROND;
      TERM;
BA3   COLCT,INT(1),BLD641 TURNAROND;
      TERM;
      ENDNETWORK;
FIN;
%EOR
 &CONFIG ISTOP=2,LOGNL=.F.,VARNL=.F. &END
```

APPENDIX I

NETWORK MODEL 'DISCRETE' SOURCE CODE LISTING

```
C*************************************************************************
C                                                                        *
C  Name: NETWORK (MAIN)                                                  *
C  Module Number: 1.0                                                    *
C  Function: Sets the NSET/QSET size and calls SLAM.                     *
C  Attributes Referenced: none                                           *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: none                                    *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: SLAM                                                  *
C  Calling Modules: N/A                                                  *
C  Scheduled by: N/A                                                     *
C                                                                        *
C*************************************************************************
C
      PROGRAM MAIN(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      DIMENSION NSET(20000)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON QSET(20000)
      EQUIVALENCE (NSET(1),QSET(1))
      NNSET=20000
      NCRDR=5
      NPRNT=6
      NTAPE=7
      CALL SLAM
      END
C*************************************************************************
C                                                                        *
C  Name: ARVL                                                            *
C  Module Number: 1.E.5                                                  *
C  Function: Sets all new arrivals attributes and routes to either       *
C            interactive or batch network arrivals.                      *
C  Attributes Referenced: ILOCAT,IMACH                                   *
C  Global (XX) Variables Referenced: IHOUR,UPTTIM                        *
C  Common Blocks/Variables Used: DIMEN/LOGFLG                            *
C    ENTCOD/IBATCH                                                       *
C    EVTCOD/IARVEV                                                       *
C    INVALS/ARRATE                                                       *
C    OFFSET/IBATST                                                       *
C  Common Blocks/Variables Changed: none                                 *
C                                                                        *
C  Modules Called: ARVTIM,ENTER,ENTINT,GETARV.LOG,SCHDL                  *
C  Calling Modules: EVENT                                                *
C  Scheduled by: ARVL,INTLC                                              *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE ARVL
C
C**** COMMON BLOCKS
      PARAMETER (MAXLOC=10,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
```

```
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/ENTCOD/ INTARV,IBATCH,NRLSIN,IPRTEN
      COMMON/EVTCOD/ ICLKEV,INTHEV,IARVEV,IPEREV,ICPDEV,LATEVT,IRESEV,
     +  ISTCEV
      COMMON/INVALS/ ARRATE(MAXLOC,24),DIALUP(MAXCPU,24),FRTABL(MAXLOC,
     + MAXCPU,MAXCLS,MAXSIZ)
      COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
     +  IPORTS
      EQUIVALENCE (ILOCAT,A(2)),(IMACH,A(3))
      EQUIVALENCE (IHOUR,XX(1)),(UPTTIM,XX(5))
C
C**** SCHEDULE NEXT ARRIVAL
      IORGM=IMACH
      IF (ARRATE(ILOCAT,IHOUR) .GT. 0.0) THEN
          IMACH=0
          XTIME=ARVTIM(ARRATE(ILOCAT,IHOUR))
      ELSE
          IMACH=1
          XTIME=UPTTIM-TNOW
      ENDIF
      CALL SCHDL(IARVEV,XTIME,A)
C
C**** IF ONLY CHECK FOR UPDATE THEN RETURN
      IF (IORGM .GT. 0) THEN
          IF (LOGFLG) CALL LOG(1,-1)
          RETURN
      ENDIF
C
C**** SET ARRIVAL ATTRIBUTES
      CALL GETARV
C
C**** SUBMIT TO EITHER INTERACTIVE ARRIVAL OR BATCH ARRIVAL
      IF (ILOCAT .LT. IBATST) THEN
          CALL ENTINT
      ELSE
          CALL ENTER(IBATCH,A)
      ENDIF
      IF (LOGFLG) CALL LOG(2,-1)
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: BLOCK DATA                                                      *
C  Module Number: 1.B                                                    *
C  Function: Defines model's common blocks and sets defaults via         *
C            DATA statements.                                            *
C  Attributes Referenced: none                                           *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: All                                     *
C  Common Blocks/Variables Changed: See DATA statements                  *
C  Modules Called: N/A                                                   *
C  Calling Modules: N/A                                                  *
C  Scheduled by: N/A                                                     *
```

```
C                                                                       *
C***********************************************************************
C
      BLOCK DATA
C
      PARAMETER (MAXLOC=10,MAXCPU=6,MAXCLS=6,MAXSIZ=6,MAXPRT=10)
      PARAMETER (MAXGAT=4,MAXWAT=5)
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/ANYVAL/ ANYDST(MAXCLS,MAXCPU)
      COMMON/CPSTAT/ ITRMOL,ITRMWP,ITRMAC,IEXJOB,IMEM,ICPU,INOUT,IMPL,
     +  CNVCPU(MAXCPU)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/ENTCOD/ INTARV,IBATCH,NRLSIN,IPRTEN
      COMMON/EVTCOD/ ICLKEV,INTHEV,IARVEV,IPEREV,ICPDEV,LATEVT,IRESEV,
     +  ISTCEV
      COMMON/FACTOR/ DISTRB(MAXCLS,MAXSIZ,6,2)
      COMMON/INVALS/ ARRATE(MAXLOC,24),DIALUP(MAXCPU,24),FRTABL(MAXLOC,
     + MAXCPU,MAXCLS,MAXSIZ)
      COMMON/INTPRB/ ONEJOB,WPPROB(MAXCLS),BATINT(MAXCPU,MAXCLS)
      COMMON/LOGPRM/ LOGON(25),LOGCNT(25)
      COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
     +  IPORTS
      COMMON/PRTPRM/ PRTSPD(MAXPRT),DUPRT,PRTTAB(MAXCPU,MAXLOC,MAXPRT),
     +  PCPRTM,PCPRTS
      COMMON/SCHEDL/ SCHBAT(MAXGAT,7,2),SCHCPU(MAXCPU,7,2),
     +  IPRGAT(MAXGAT,MAXPRT),ICRGAT(MAXCPU,MAXGAT)
      COMMON/UNITS/ IFACT,ISTUD,IADMN,IOTHR,IBTCH,ICONST
      COMMON/VARCOM/ CPULTM,CPULTS,TRMLTM,TRMLTS,OPRMN,OPRSTD,XLOGMN,
     +  XLOGST
      COMMON/WAITCM/ TRMWAT(MAXLOC,MAXWAT),WAITMN,WAITST,PORTWT(MAXWAT)
C
C**** DATA STATEMENTS
      DATA ICLKEV,INTHEV,IARVEV,IPEREV,ICPDEV,LATEVT/1,2,5,6,8,10/
      DATA IRESEV,ISTCEV/3,4/
      DATA ITRMOL,ITRMWP,ITRMAC,IEXJOB,IMEM,ICPU,INOUT,IMPL/1,2,3,4,5,
     +  6,7,8/
      DATA INTARV,IBATCH,NRLSIN,IPRTEN/1,2,3,4/
      DATA LOGON/25*0/
      DATA LOGCNT/25*0/
      DATA ICPUOS,IHLDOS,IRTSTR,ITRMOS,IGATOS,IPRTOS/9,0,1,0,0,7/
      DATA IPORTS/39/
      DATA SCHBAT,SCHCPU/140*-1.0/
      DATA IFACT,ISTUD,IADMN,IOTHR,IBTCH,ICONST/8,9,10,11,12,13/
      DATA CPULTM,CPULTS/28800.,7200./
      DATA TRMLTM,TRMLTS/7200.,3600./
      DATA OPRMN,OPRSTD,XLOGMN,XLOGST/1500.,300.,60.,30./
      DATA PORTWT/0.8,0.5,0.4,0.2,0.1/
      DATA WAITMN,WAITST/7200.,3600./
      DATA PCPRTM,PCPRTS/500.,200./
      DATA IPRGAT/40*0/
      DATA ICRGAT/24*0/
      DATA CNVCPU/6*1.0/
```

```
      END
C*************************************************************************
C                                                                        *
C  Name: CALSCH                                                          *
C  Module Number: 1.C.1                                                  *
C  Function: Schedules all computers up/down and batch locations         *
C            open/close for a day.                                       *
C  Attributes Referenced: ICOMP,IGATE,ITYPE                              *
C  Global (XX) Variables Referenced: IDAY                                *
C  Common Blocks/Variables Used: DIMEN/NUMCPU,NUMGAT                     *
C    EVTCOD/IRESEV                                                       *
C    OFFSET/ICPUOS,IGATOS                                                *
C    SCHEDL/SCHCPU,SCHBAT                                                *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: SCHDL                                                 *
C  Calling Modules: DYPROC,INTLC                                         *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE CALSCH
      PARAMETER (MAXGAT=4,MAXCPU=6,MAXPRT=10)
C
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/EVTCOD/ ICLKEV,INTHEV,IARVEV,IPEREV,ICPDEV,LATEVT,IRESEV,
     +  ISTCEV
      COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
     +  IPORTS
      COMMON/SCHEDL/ SCHBAT(MAXGAT,7,2),SCHCPU(MAXCPU,7,2),
     +  IPRGAT(MAXGAT,MAXPRT),ICRGAT(MAXCPU,MAXGAT)
      EQUIVALENCE (IDAY,XX(2))
      EQUIVALENCE (ICOMP,IGATE,A(2)),(ITYPE,A(3))
C
C**** SCHEDULE COMPUTERS UP/DOWN TODAY
      DO 100  I=1,NUMCPU
          ICOMP=ICPUOS+I
          ITYPE=0
          IF (SCHCPU(I,IDAY,2) .GE. 0.0) CALL SCHDL(IRESEV,
     +        SCHCPU(I,IDAY,2),A)
          ITYPE=1
          IF (SCHCPU(I,IDAY,1) .GE. 0.0) CALL SCHDL(IRESEV,
     +        SCHCPU(I,IDAY,1),A)
  100 CONTINUE
C
C**** SCHEDULE BATCH(GATES) OPEN/CLOSE TODAY
      DO 200 I=1,NUMGAT
          IGATE=IGATOS+I
          ITYPE=0
          IF (SCHBAT(I,IDAY,2) .GE. 0.0) CALL SCHDL(IRESEV,
     +        SCHBAT(I,IDAY,2),A)
          ITYPE=1
          IF (SCHBAT(I,IDAY,1) .GE. 0.0) CALL SCHDL(IRESEV,
```

```
     +        SCHBAT(I,IDAY,1),A)
  200 CONTINUE
      RETURN
      END
C*********************************************************************
C                                                                     *
C  Name: CLOCK                                                        *
C  Module Number: 1.E.1                                               *
C  Function: Controls all time processing-- end of hour, end of day,  *
C            and end of week.                                         *
C  Attributes Referenced: none                                        *
C  Global (XX) Variables Referenced: IHOUR,IDAY,IWEEK,ISTOP           *
C  Common Blocks/Variables Used: EVTCOD/ICLKEV                        *
C  Common Blocks/Variables Changed: none                              *
C  Modules Called: DYPROC,EDPROC,HRPROC,SCHDL,WKPROC                  *
C  Calling Modules: EVENT                                             *
C  Scheduled by: CLOCK,INTLC                                          *
C                                                                     *
C*********************************************************************
C
      SUBROUTINE CLOCK
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/EVTCOD/ ICLKEV,INTHEV,IARVEV,IPEREV,ICPUEV,LATEVT,IRESEV,
     +  ISTCEV
C
      EQUIVALENCE (IHOUR,XX(1)),(IDAY,XX(2)),(IWEEK,XX(3)),
     +  (ISTOP,XX(6))
C
C**** SCHEDULE NEXT UPDATE
      CALL SCHDL(ICLKEV,3600.,A)
C**** DO END HOUR PROCESSING
      IHOUR=IHOUR+1
      CALL HRPROC
C**** IF END OF DAY DO END DAY PROCESSING
      IF (IHOUR .EQ. 25) THEN
          IDAY=IDAY+1
          IHOUR=1
          CALL DYPROC
      ENDIF
C**** IF END OF WEEK DO END WEEK PROCESSING
      IF (IDAY .EQ. 8) THEN
          IDAY=1
          IWEEK=IWEEK+1
          CALL WKPROC
      ENDIF
C**** CHECK FOR END OF RUN
      IF (IWEEK .EQ. ISTOP) CALL EDPROC
      RETURN
      END
C*********************************************************************
C                                                                     *
C  Name: CPUARV                                                       *
```

```
C  Module Number: 1.E.9                                                  *
C  Function: Calculates a job's run parameters; adjusts CPU's           *
C            states; schedules departure if room in 'execute' else      *
C            puts job in hold queue.                                    *
C  Attributes Referenced: ILOCAT,IMACH,ICLASS,ISIZE,AWAIT,ASECCP,       *
C    ARUN,INTBAT,IORG,AMEM,AIOSEC                                       *
C  Global (XX) Variables Referenced: none                               *
C  Common Blocks/Variables Used: CPSTAT/CNVCPU,ITRMAC,IMPL,IEXJOB,      *
C      ICPU,IMEM,INOUT                                                  *
C    DIMEN/LOGFLG                                                       *
C    EVTCOD/ICPDEV                                                      *
C    OFFSET/IBATST,IHLDOS                                               *
C  Common Blocks/Variables Changed: SS area                             *
C  Modules Called: CPUSEC,FIGRUN,FILEM,LOG,MEMSIZ,SCHDL,XIOSEC          *
C  Calling Modules: EVENT,INTHND                                        *
C  Scheduled by: NETWORK-BATCH ARRIVALS                                 *
C                                                                       *
C***********************************************************************
C
      SUBROUTINE CPUARV
      REAL MEMSIZ
      PARAMETER (MAXCPU=6)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/CPSTAT/ ITRMOL,ITRMWP,ITRMAC,IEXJOB,IMEM,ICPU,INOUT,IMPL,
     +  CNVCPU(MAXCPU)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/EVTCOD/ ICLKEV,INTHEV,IARVEV,IPEREV,ICPDEV,LATEVT,IRESEV,
     +  ISTCEV
      COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
     + IPORTS
      EQUIVALENCE (ILOCAT,A(2)),(IMACH,A(3)),(ICLASS,A(4)),(ISIZE,A(5)),
     +  (AWAIT,A(6)),(ASECCP,A(7)),(ARUN,A(8)),(INTBAT,A(9)),
     +  (IORG,A(12)),(AMEM,A(14)),(AIOSEC,A(15))
C
C**** FIGURE CPU TIME, I/O TIME, AND MEMORY SIZE
      ASECCP=CPUSEC(IORG,ISIZE)*CNVCPU(IMACH)
      AMEM=MEMSIZ(IORG,ISIZE)
      AIOSEC=XIOSEC(IORG,ISIZE)
C
C**** FIGURE COMPUTER INDEX
      INDEX=IMACH*10-10
C
C**** IF IN 'WAIT' MODE WHILE INTERACTIVE - DECREMENT ACTIVE TERMINAL
      IF ((INTBAT .EQ. 1).AND.(ILOCAT .LT. IBATST))
     +   SS(INDEX+ITRMAC)=SS(INDEX+ITRMAC)-1.
      IF (SS(INDEX+IMPL) .LE. SS(INDEX+IEXJOB)) THEN
C         *** NO ROOM - PUT IN HOLD QUEUE
          AWAIT=TNOW
          IF (LOGFLG) CALL LOG(3,INDEX)
          CALL FILEM(IMACH+IHLDOS,A)
      ELSE
```

```
C          *** START THE JOB ***
           ARUN=FIGRUN(IMACH,IORG,ASECCP,AMEM)
           SS(INDEX+IEXJOB)=SS(INDEX+IEXJOB)+1.
           SS(INDEX+ICPU)=SS(INDEX+ICPU)+ASECCP
           SS(INDEX+IMEM)=SS(INDEX+IMEM)+AMEM
           SS(INDEX+INOUT)=SS(INDEX+INOUT)+AIOSEC
           IF (LOGFLG) CALL LOG(4,INDEX)
C          **** SCHEDULE DEPARTURE ****
           CALL SCHDL(ICPDEV,ARUN,A)
         ENDIF
         RETURN
         END
C*************************************************************************
C                                                                        *
C  Name: CPUDPT                                                          *
C  Module Number: 1.E.8                                                  *
C  Function: Adjust CPU states as job leaves 'execute'; routes to        *
C            printer if word processing; and if job in hold queue -      *
C            start.                                                      *
C  Attributes Referenced: ILOCAT,IMACH,ICLASS,ISIZE,APRTQ,ASECCP,        *
C    AMACH,ARUN,APRTL,INTBAT,IORG,AMEM,ALOCAT,AIOSEC                     *
C  Global (XX) Variables Referenced: IWPCLS                              *
C  Common Blocks/Variables Used: CPSTAT/IEXJOB,IMEM,ICPU,INOUT           *
C    DIMEN/LOGFLG                                                        *
C    EVTCOD/ICPDEV                                                       *
C    OFFSET/IBATST,IRTSTR,IHLDOS                                         *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: COLCT,FIGRUN,INTHND,LOG,PRTLIN,RMOVE,RTPRT,SCHDL      *
C  Calling Modules: EVENT                                                *
C  Scheduled by: CPUARV,CPUDPT                                           *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE CPUDPT
      PARAMETER (MAXCPU=6)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/CPSTAT/ ITRMOL,ITRMMP,ITRMAC,IEXJOB,IMEM,ICPU,INOUT,IMPL,
     +  CNVCPU(MAXCPU)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/EVTCOD/ ICLKEV,INTHEV,IARVEV,IPEREV,ICPDEV,LATEVT,IRESEV,
     +  ISTCEV
      COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
     +  IPORTS
      EQUIVALENCE (ILOCAT,A(2)),(IMACH,A(3)),(ICLASS,A(4)),(ISIZE,A(5)),
     +  (APRTQ,A(6)),(ASECCP,AMACH,A(7)),(ARUN,APRTL,A(8)),
     +  (INTBAT,A(9)),(IORG,A(12)),(AMEM,ALOCAT,A(14)),(AIOSEC,A(15))
      EQUIVALENCE (IWPCLS,XX(4))
C
C**** CHANGE CPU STATES
      INDEX=IMACH*10-10
      SS(INDEX+IEXJOB)=SS(INDEX+IEXJOB)-1.
```

```
      SS(INDEX+IMEM)=SS(INDEX+IMEM)-AMEM
C
C**** COLLECT INTERACTIVE RESPONSE TIMES
      IF ((ILOCAT .LT. IBATST).AND.(INTBAT .EQ. 0)) THEN
          I=IRTSTR+(IMACH-1)
          CALL COLCT(ARUN,I)
      ENDIF
C
C**** INPUT TO PRINTER IF BATCH OR WORD PROCESSING CLASS
      IF ((ILOCAT .GE. IBATST) .OR. (ICLASS .EQ. IWPCLS)) THEN
          ALOCAT=REAL(ILOCAT)
          AMACH=REAL(IMACH)
          APRTL=PRTLIN(IORG,ISIZE)
          APRTQ=TNOW
          CALL RTPRT(IMACH,ILOCAT)
      ELSE
C         **** ROUTE TO INTHND IF IN WAIT MODE AND CPU UP ****
          IF ((INTBAT .EQ. 1).AND.(XX(IMACH+ICPUOS).GT.0.5)) CALL INTHND
      ENDIF
      IF (LOGFLG) CALL LOG(5,INDEX)
C
C**** IF ANY JOBS IN HOLD QUEUE - START ONE
      IF (NNQ(IMACH+IHLDOS) .GT. 0) THEN
          CALL RMOVE(1,IMACH+IHLDOS,A)
          INDEX=IMACH*10-10
          SS(INDEX+IEXJOB)=SS(INDEX+IEXJOB)+1.
          SS(INDEX+ICPU)=SS(INDEX+ICPU)+ASECCP
          SS(INDEX+IMEM)=SS(INDEX+IMEM)+AMEM
          SS(INDEX+INOUT)=SS(INDEX+INOUT)+AIOSEC
          ARUN=FIGRUN(IMACH,IORG,ASECCP,AMEM)
          IF (LOGFLG) CALL LOG(6,INDEX)
          CALL SCHDL(ICPDEV,ARUN,A)
      ENDIF
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: DYPROC                                                          *
C  Module Number: 1.E.1.2                                                *
C  Function: Performs end of day processing -- calculates schedule       *
C            and updates arrival rates.                                  *
C  Attributes Referenced: none                                           *
C  Global (XX) Variables Referenced: IDAY                                *
C  Common Blocks/Variables Used: DIMEN/LOGFLG                            *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: CALSCH,UPDARV                                         *
C  Calling Modules: CLOCK                                                *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE DYPROC
C
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
```

```
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      EQUIVALENCE (IDAY,XX(2))
C
C**** UPDATE ARRIVAL RATES IF NOT END OF WEEK
      IF (IDAY .LE. 7) THEN
          CALL UPDARV
          CALL CALSCH
      ENDIF
      IF (LOGFLG) WRITE(NPRNT,10) TNOW,IDAY
      RETURN
   10 FORMAT('0ARRIVAL TABLES UPDATE ',E11.4,'  DAY = ',I3)
      END
C*************************************************************************
C                                                                        *
C  Name: EDPROC                                                          *
C  Module Number: 1.E.1.4                                                *
C  Function: Performs end of run processing -- csets MSTOP=-1            *
C  Attributes Referenced: none                                           *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: none                                    *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: none                                                  *
C  Calling Modules: CLOCK                                                *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE EDPROC
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
C
C
      MSTOP=-1
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: ENTINT                                                          *
C  Module Number: 1.C.2                                                  *
C  Function: Checks resource availability for an interactive             *
C            arrival and may enter into network.                         *
C  Attributes Referenced: ILOCAT,IMACH                                   *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: DIMEN/LOGFLG                            *
C     ENTCOD/INTARV                                                      *
C     OFFSET/ICPUOS,ITRMOS                                               *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: ENTER,LOG,RESCHD,WAITTR                               *
C  Calling Modules: ARVL,LATARV,PERIOD                                   *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
```

```
C
      SUBROUTINE ENTINT
C
C**** COMMON BLOCKS
      PARAMETER (MAXCLS=6,MAXSIZ=6)
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/ENTCOD/ INTARV,IBATCH,NRLSIN,IPRTEN
      COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
     +  IPORTS
      EQUIVALENCE (ILOCAT,A(2)),(IMACH,A(3))
C
C**** IS COMPUTER AVAILABLE
      IF (XX(IMACH+ICPUOS) .GT. 0.5) THEN
C         **** YES - BUT IS A TERMINAL AVAILABLE ****
          IF (NNRSC(ILOCAT+ITRMOS) .GT. 0) THEN
C             **** YES - GRAB IT ****
              CALL ENTER(INTARV,A)
          ELSE
C             **** NO - WILL THE USER WAIT? ****
              X=WAITTR(DUMMY)
              IF (X .GT. 0.5) THEN
C                 **** YES - ENTER TERMINAL QUEUE ****
                  CALL ENTER(INTARV,A)
              ENDIF
          ENDIF
      ELSE
C         **** RESCHEDULE LATER ARRIVAL ****
          CALL RESCHD
      ENDIF
C
      IF (LOGFLG) CALL LOG(19,-1)
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: EVENT                                                           *
C  Module Number: 1.2                                                    *
C  Function: Calls approparate sub-routine as events exit event          *
C            calandar.                                                   *
C  Attributes Referenced: none                                           *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: none                                    *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: ARVL,CLOCK,CPUARV,CPUDPT,INTHND,LATARV,LOGON,         *
C                  PERIOD,RESORC,SETCPU                                  *
C  Calling Modules: SLAM                                                 *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE EVENT(KODE)
C
```

```
      GO TO (10,20,30,40,50,60,70,80,90,100) KODE
   10 CALL CLOCK
      RETURN
   20 CALL INTHND
      RETURN
   30 CALL RESORC
      RETURN
   40 CALL SETCPU
      RETURN
   50 CALL ARVL
      RETURN
   60 CALL PERIOD
      RETURN
   70 CALL LOGON
      RETURN
   80 CALL CPUDPT
      RETURN
   90 CALL CPUARV
      RETURN
  100 CALL LATARV
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: FIGUTL                                                          *
C  Module Number: 1.E.1.1.1                                              *
C  Function: Figures and prints CPU and I/O utilizations.                *
C  Attributes Referenced: none                                           *
C  Global (XX) Variables Referenced: INTER                               *
C  Common Blocks/Variables Used: CPSTAT/ICPU,INOUT                       *
C    DIMEN/NUMCPU                                                        *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: none                                                  *
C  Calling Modules: HRPROC                                               *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE FIGUTL
      PARAMETER (MAXCPU=6)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/CPSTAT/ ITRMOL,ITRMWP,ITRMAC,IEXJOB,IMEM,ICPU,INOUT,IMPL,
     +  CNVCPU(MAXCPU)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      EQUIVALENCE (INTER,XX(19))
C
C**** FIGURE CPU AND I/O UTILIZATIONS
      XINTER=REAL(INTER)*3600.
      WRITE(NPRNT,500)
      DO 100  I=1,NUMCPU
          INDEX=I*10-10
```

```
            UTILC=SS(INDEX+ICPU)/XINTER
            UTILI=SS(INDEX+INOUT)/XINTER
            WRITE(NPRNT,510) I,UTILC,UTILI
            SS(INDEX+ICPU)=0.0
            SS(INDEX+INOUT)=0.0
  100 CONTINUE
      RETURN
C
C**** FORMAT STATEMENTS
  500 FORMAT('0**** CPU AND I/O UTILIZATIONS ****',/,
     +  'COMPUTER  CPU UTIL  I/O UTIL')
  510 FORMAT(' ',I4,F13.4,F10.4)
      END
C*************************************************************************
C                                                                        *
C  Name: GETARV                                                          *
C  Module Number: 1.E.5.1                                                *
C  Function: Sets arrival attributes.                                    *
C  Attributes Referenced: ATIME,ILOCAT,IMACH,ICLASS,ISIZE,AWAIT,         *
C    ALOCJP,ACPU,INTBAT,IDIAL,IORG                                       *
C  Global (XX) Variables Referenced: IHOUR                               *
C  Common Blocks/Variables Used: ANYVAL/ANYDST                           *
C    DIMEN/NUMCPU,NUMCLS,NUMSIZ                                          *
C    INVALS/DIALUP,FRTABL                                                *
C    OFFSET/IBATST                                                       *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: none                                                  *
C  Calling Modules: ARVLT                                                *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE GETARV
      PARAMETER (MAXLOC=10,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/ANYVAL/ ANYDST(MAXCLS,MAXCPU)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/INVALS/ ARRATE(MAXLOC,24),DIALUP(MAXCPU,24),FRTABL(MAXLOC,
     + MAXCPU,MAXCLS,MAXSIZ)
      COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
     +  IPORTS
      EQUIVALENCE (ATIME,A(1)),(ILOCAT,A(2)),(IMACH,A(3)),(ICLASS,A(4)),
     +  (ISIZE,A(5)),(AWAIT,A(6)),(ALOCJP,A(7)),(ACPU,A(8)),
     +  (INTBAT,A(9)),(IDIAL,A(11)),(IORG,A(12))
      EQUIVALENCE (IHOUR,XX(1))
C
C**** SET ARRIVAL ATTRIBUTES
      ATIME=TNOW
      AWAIT=0.0
      RN=DRAND(1)
      DO 500  I=1,NUMCPU+1
```

```
            DO 400  J=1,NUMCLS
                DO 300  K=1,NUMSIZ
                    IF (RN .LE. FRTABL(ILOCAT,I,J,K)) THEN
                        IMACH=I
                        ICLASS=J
                        ISIZE=K
                        GO TO 525
                    ENDIF
  300           CONTINUE
  400       CONTINUE
  500 CONTINUE
C
C**** IF MACHINE ='ANY' THEN DETERMINE MACHINE
  525 IF (IMACH .GE. NUMCPU+1) THEN
            RN=DRAND(1)
            DO 550  I=1,NUMCPU
                IF (RN .LE. ANYDST(ICLASS,I)) THEN
                    IMACH=I
                    GO  TO 600
                ENDIF
  550       CONTINUE
        ENDIF
  600 IORG=ICLASS
        INTBAT=0
C
C**** IS THIS A DIALUP ARRIVAL
        RN=DRAND(1)
        IDIAL=0
        ACPU=REAL(IMACH)
        ALOCJP=REAL(ILOCAT)
        IF ((ILOCAT .LT. IBATST).AND.(RN .LE. DIALUP(IMACH,IHOUR))) THEN
             IDIAL=1
             ALOCJP=0.0
        ENDIF
C
        RETURN
        END
C*************************************************************************
C                                                                        *
C  Name: HRPROC                                                          *
C  Module Number: 1.E.1.1                                                *
C  Function: Performs end of hour processing.                            *
C  Attributes Referenced: none                                           *
C  Global (XX) Variables Referenced: IHOUR,IDAY,IWEEK,UPTTIM,XLOGON,*
C    XLOGOF,ISUM,INTER,ITOTAL                                            *
C  Common Blocks/Variables Used: DIMEN/LOGFLG                            *
C    LOGPRM/LOGCNT                                                       *
C  Common Blocks/Variables Changed: DIMEN/LOGFLG                         *
C    LOGPRM/LOGCNT                                                       *
C  Modules Called: CLEAR,FIGUTL,SUMRY                                    *
C  Calling Modules: CLOCK                                                *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
```

```
      SUBROUTINE HRPROC
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/LOGPRM/ LOGON(25),LOGCNT(25)
      EQUIVALENCE (IHOUR,XX(1)),(IDAY,XX(2)),(IWEEK,XX(3)),
     +  (UPTTIM,XX(5)),(XLOGON,XX(7)),(XLOGOF,XX(8)),(ISUM,XX(17)),
     +  (INTER,XX(19)),(ITOTAL,XX(20))
C
C**** INCREMENT UPDATE TIME AND STATISTICS INTERVAL
      UPTTIM=UPTTIM+3600.
      ITOTAL=ITOTAL+1
C
C**** CHECK FOR LOG ON/OFF
      IF (TNOW .GT. XLOGOF) THEN
          LOGFLG=.FALSE.
      ELSE
          IF (TNOW .GT. XLOGON) LOGFLG=.TRUE.
      ENDIF
C
C**** IF INTERVAL REACHED - PRINT STATS AND CLEAR
      IF (ITOTAL .EQ. INTER) THEN
          ITOTAL=0
          IF (ISUM .EQ. 1) THEN
              CALL SUMRY
              WRITE(NPRNT,510) (LOGCNT(I), I=1,25)
              DO 100  I=1,25
                  LOGCNT(I)=0
  100         CONTINUE
              WRITE(NPRNT,500) IWEEK,IDAY,IHOUR-1
              CALL FIGUTL
          ENDIF
          CALL CLEAR
      ENDIF
      RETURN
  500 FORMAT(' ','WEEK NUMBER: ',I2,'  DAY: ',I2,'  HOUR: ',I3)
  510 FORMAT('0',12I10,/,1X,12I10)
      END
C*************************************************************************
C                                                                        *
C  Name: INTHND                                                          *
C  Module Number: 1.E.2                                                  *
C  Function: Handles interactive sessions by routing jobs to CPU         *
C            and scheduling think times.                                 *
C  Attributes Referenced: IMACH,ICLASS,INTBAT,IJOBS,IORG,AMEAN           *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: CPSTAT/ITRMAC                           *
C    DIMEN/LOGFLG                                                        *
C    EVTCOD/INTHEV                                                       *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: CPUARV,LOG,SCHDL,SESOVR,SETINT                        *
C  Calling Modules: EVENT,CPUDPT                                         *
```

```
C  Scheduled by: INTHND,RTPRT,SETCPU,NETWORK-INTERACTIVE ARRIVALS   *
C                                                                   *
C*******************************************************************
C
      SUBROUTINE INTHND
      PARAMETER (MAXCPU=6)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      LOGICAL LOGFLG
      COMMON/CPSTAT/ ITRMOL,ITRMWP,ITRMAC,IEXJOB,IMEM,ICPU,INOUT,IMPL,
     +  CNVCPU(MAXCPU)
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSI2,NUMPRT,NUMGAT,LOGFLG
      COMMON/EVTCOD/ ICLKEV,INTHEV,IARVEV,IPEREV,ICPDEV,LATEVT,IRESEV,
     +  ISTCEV
      EQUIVALENCE (IMACH,A(3)),(ICLASS,A(4)),(INTBAT,A(9)),
     +  (IJOBS,A(10)),(IORG,A(12)),(AMEAN,A(13))
C
C**** IF JOB JUST FINISHED CPU OR PRINTER IN WAIT MODE THEN SCHEDULE
C**** THINK TIME
C
      IF (INTBAT .EQ. 1) THEN
          INDEX=IMACH*10-10
          SS(INDEX+ITRMAC)=SS(INDEX+ITRMAC)+1.
          IF (IJOBS .GT. 0) THEN
              INTBAT=0
              CALL SCHDL(INTHEV,EXPON(AMEAN,1),A)
          ELSE
              CALL SESOVR
          ENDIF
      ELSE
C         **** START ANOTHER JOB ****
          IJOBS=IJOBS-1
          ICLASS=IORG
          CALL SETINT
          CALL CPUARV
C         **** IF BATCH MODE AND JOBS LEFT THEN SCHEDULE THINK TIME ****
          IF (INTBAT .EQ. 0) THEN
              IF (IJOBS .GT. 0) THEN
                  CALL SCHDL(INTHEV,EXPON(AMEAN,1),A)
              ELSE
C                 **** SESSION OVER ****
                  CALL SESOVR
              ENDIF
          ENDIF
      ENDIF
      IF (LOGFLG) CALL LOG(17,-1)
      RETURN
      END
C*******************************************************************
C                                                                   *
C  Name: INTLC                                                      *
C  Module Number: 1.1                                               *
C  Function: Initializes all global variables; schedules first      *
```

```
C            arrivals all locations; and intializes arrival rates   *
C            and frequency distributions for all locations.         *
C  Attributes Referenced: ILOCAT,IMACH,ICLASS,ISIZE,ASECCP,APRNT,   *
C    INTBAT,ALOCJP,IDIAL,IORG,AMEAN,AMEM                             *
C  Global (XX) Variables Referenced: IHOUR,IDAY,IWEEK,IWPCLS,UPTTIM,*
C    ISTOP,XLOGON,XLOGOF,ISUM,INTER,ITOTAL                           *
C  Common Blocks/Variables Used: DIMEN/NUMCPU,NUMCLS,NUMSIZ,NUMLOC  *
C    EVTCOD/ICLKEV,IARVEV,IPEREV                                     *
C    INVALS/ARRATE                                                   *
C    UNITS/ISTUD,IBTCH,ICONST                                        *
C  Common Blocks/Variables Changed: none                             *
C  Modules Called: ARVTIM,CALSCH,LOGERR,PRTCOM,RDERR,RDPARM,SCHDL,   *
C                  UPDARV,UPDFRQ                                     *
C  Calling Modules: SLAM                                             *
C  Scheduled by: none                                                *
C                                                                    *
C********************************************************************
C
      SUBROUTINE INTLC
      PARAMETER (MAXLOC=10,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/EVTCOD/ ICLKEV,INTHEV,IARVEV,IPEREV,ICPDEV,LATEVT,IRESEV,
     +   ISTCEV
      COMMON/INVALS/ ARRATE(MAXLOC,24),DIALUP(MAXCPU,24 ,FRTABL(MAXLOC,
     + MAXCPU,MAXCLS,MAXSIZ)
      COMMON/UNITS/ IFACT,ISTUD,IADMN,IOTHR,IBTCH,ICONST
      EQUIVALENCE (ILOCAT,A(2)),(IMACH,A(3)),(ICLASS,A(4)),(ISIZE,A(5)),
     +   (ASECCP,ALOCJP,A(7)),(APRNT,A(8)),(INTBA ,A(9)),
     +   (IDIAL,A(11)),(IORG,A(12)),(AMEAN,A(13)),(AMEM,A(14))
      EQUIVALENCE (IHOUR,XX(1)),(IDAY,XX(2)),(IWEEK,XX(3)),
     +   (IWPCLS,XX(4)),(UPTTIM,XX(5)),(ISTOP,XX(6)),(XLOGON,XX(7)),
     +   (XLOGOF,XX(8)),(ISUM,XX(17)),(INTER,XX(19)),(ITOTAL,XX(20))
C
C**** INITIALIZE XX AND SS ARRAY TO 0.0
      DO 25  I=1,100
          XX(I)=0.0
          SS(I)=0.0
   25 CONTINUE
C
C**** SET DEFAULTS XX PARAMETERS
      IHOUR=1
      IDAY=1
      ITOTAL=0
      IWEEK=1
      UPTTIM=3600.1
      ISTOP=11
      XLOGON=1.0
      XLOGOF=1.0
      ISUM=1
      INTER=6
```

```
C
C**** READ IN ALL THE PARAMETERS
      CALL RDPARM
      CALL PRTCON
C
C**** INIT ARRIVAL AND FRE.UENCY TABLES
      DO 225  I=ISTUD,IBTCH
          READ(I) JJ,LL,MM
          IF ((JJ.NE.NUMCPU+1).OR.(LL.NE.NUMCLS).OR.(MM.NE.NUMSIZ)) THEN
              CALL LOGERR(2)
              STOP
          ENDIF
  225 CONTINUE
      DO 275  I=1,IWEEK
          CALL UPDFRQ
          IF (I .NE. IWEEK) THEN
              DO 250  J=1,7
                  CALL UPDARV
  250         CONTINUE
          ELSE
              CALL UPDARV
          ENDIF
  275 CONTINUE
C
C**** SET TIME AND SCHEDULE FIRST CLOCK UPDATE
      STIME=REAL(IWEEK-1)
      IF (STIME .GT. 0.0) TNOW=STIME*86400.+7.
      CALL SCHDL(ICLKEV,3600.,A)
C
C**** SCHEDULE FIRST ARRIVALS AT ALL LOCATIONS
      DO 300  I=1,NUMLOC
          ILOCAT=I
          IF (ARRATE(I,1) .NE. 0.0) THEN
              IMACH=0
              XTIME=ARVTIM(ARRATE(I,1))
          ELSE
              IMACH=1
              XTIME=3600.1
          ENDIF
          CALL SCHDL(IARVEV,XTIME,A)
  300 CONTINUE
C
C**** SCHEDULE COMPUTERS AND BATCH LOCATIONS UP/DOWN
      CALL CALSCH
C
C**** SCHEDULE FIRST PERIODIC ARRIVAL
  400 READ(ICONST,ERR=450) XTIME,ILOCAT,IMACH,ICLASS,ISIZE
      IF (XTIME .LT. TNOW) GO TO 400
      INTBAT=0
      IDIAL=0
      IORG=ICLASS
      ALOCJP=REAL(ILOCAT)
      CALL SCHDL(IPEREV,XTIME,A)
      RETURN
  450 CALL RDERR(ICONST)
```

```
      END
C*********************************************************************
C                                                                     *
C  Name: LATARV                                                       *
C  Module Number: 1.E.10                                              *
C  Function: Enters the rescheduled arrivals into the network.        *
C  Attributes Referenced: ATIME,AWAIT                                 *
C  Global (XX) Variables Referenced: none                             *
C  Common Blocks/Variables Used: DIMEN/LOGFLG                         *
C  Common Blocks/Variables Changed: none                              *
C  Modules Called: ENTINT,LOG                                         *
C  Calling Modules: EVENT                                             *
C  Scheduled by: RESCHD                                               *
C                                                                     *
C*********************************************************************
C
      SUBROUTINE LATARV
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      EQUIVALENCE (ATIME,A(1)),(AWAIT,A(6))
C
C**** ENTER INTO INTERACTIVE ARRIVALS
      ATIME=TNOW
      AWAIT=0.0
      IF (LOGFLG) CALL LOG(8,-1)
      CALL ENTINT
      RETURN
      END
C*********************************************************************
C                                                                     *
C  Name: LOG                                                          *
C  Module Number: 1.C.3                                               *
C  Function: Logs the subroutine name and attributes.                 *
C  Attributes Referenced: none directly                               *
C  Global (XX) Variables Referenced: none                             *
C  Common Blocks/Variables Used: LOGPRM/LOGCNT,LOGON                  *
C  Common Blocks/Variables Changed: LOGPRM/LOGCNT                     *
C  Modules Called: none                                               *
C  Calling Modules: almost all, see code                              *
C  Scheduled by: none                                                 *
C                                                                     *
C*********************************************************************
C
      SUBROUTINE LOG(KODE,INDEX)
      CHARACTER LOGDSC(25)*10
      DIMENSION IA(100)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/LOGPRM/ LOGON(25),LOGCNT(25)
```

```
      EQUIVALENCE (A(1),IA(1))
C
      DATA LOGDSC/'ARVL-UPDAT','ARVL-SUBMT','CPUARV-HLD','CPUARV-RUN',
     +            'CPUDPT    ','CPUDPT-NEW','SESOVR    ','LATARV    ',
     +            'LOGON     ','PERIOD-GO ','PERIOD-SCH','RESCHD    ',
     +            'RTPRT     ','SETINT    ','WAITTR-OK ','WAITTR-LV ',
     +            'INTHND    ','RESORC    ','ENTINT    ','SETCPU    ',
     +            'STPINT    ','WAITPO-OK ','WAITPO-LV ','           ',
     +            '          '/
C
      LOGCNT(KODE)=LOGCNT(KODE)+1
      IF (LOGON(KODE) .EQ. 1) THEN
          WRITE(NPRNT,10) TNOW,LOGDSC(KODE),A(1),(IA(I), I=2,5),
     +      A(6),A(7),A(8),(IA(I), I=9,12),A(13),A(14),A(15)
          IF (INDEX .GE. 0) WRITE(NPRNT,20) (SS(INDEX+I), I=1,7)
      ENDIF
      RETURN
   10 FORMAT('0',E11.4,1X,A10,E11.4,4I4,3E11.4,4I4,3E11.4)
   20 FORMAT(' ',20X,4F5.0,2E12.4,F5.0)
      END
C*************************************************************************
C                                                                        *
C  Name: LOGERR                                                          *
C  Module Number: 1.C.4                                                  *
C  Function: Logs user errors.                                           *
C  Attributes Referenced: none                                           *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: none                                    *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: none                                                  *
C  Calling Modules: INTLC,USERF                                          *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE LOGERR(KODE)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
C
      WRITE(NPRNT,10) KODE
      WRITE(NPRNT,20) (A(I), I=1,15)
      RETURN
   10 FORMAT('1**** USER ERROR ',I2,' ****')
   20 FORMAT(' ',E11.4,4I4,3E11.4,4I4,3E11.4)
      END
C*************************************************************************
C                                                                        *
C  Name: LOGON                                                           *
C  Module Number: 1.E.7                                                  *
C  Function: Checks port availability and if available starts            *
C            session else either releases terminal or puts in wait       *
C            queue.                                                      *
C  Attributes Referenced: IMACH                                          *
```

```
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: DIMEN/LOGFLG                            *
C    ENTCOD/NRLSIN                                                        *
C    OFFSET/IPORTS                                                        *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: ENTER,FILEM,LOG,SETCPU,WAITPO                         *
C  Calling Modules: EVENT                                                *
C  Scheduled by: NETWORK-BATCH ARRIVALS                                  *
C                                                                        *
C************************************************************************
C
      SUBROUTINE LOGON
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/ENTCOD/ INTARV,IBATCH,NRLSIN,IPRTEN
      COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
     +  IPORTS
      EQUIVALENCE (IMACH,A(3))
C
C**** MAKE SURE COMPUTER STILL UP
      IF (XX(ICPUOS+IMACH) .GT. 0) THEN
C         **** IS PORT AVAILABLE?
          IF (XX(IPORTS+IMACH) .GT. 0.) THEN
C             **** YES - INPUT TO CPU INTERACTIVE HANDLER ****
              CALL SETCPU
          ELSE
C             **** NO - WILL THE USER WAIT? ****
              X=WAITPO(DUMMY)
              IF (X .LT. 0.5) THEN
C                 **** WON'T WAIT - RELEASE TERMINAL ****
                  CALL ENTER(NRLSIN,A)
              ELSE
C                 **** WILL WAIT - PUT IN QUEUE ****
                  CALL FILEM(IPORTS+IMACH,A)
              ENDIF
          ENDIF
      ELSE
C         **** COMPUTER WENT DOWN DURING LOGON
          CALL RESCHD
          CALL ENTER(NRLSIN,A)
      ENDIF
      IF (LOGFLG) CALL LOG(9,-1)
      RETURN
      END
C************************************************************************
C                                                                        *
C  Name: PERIOD                                                          *
C  Module Number: 1.E.6                                                  *
C  Function: Enters periodic arrivals into network.                      *
C  Attributes Referenced: ATIME,ILOCAT,IMACH,ICLASS,ISIZE,INTBAT,        *
C    IDIAL,ALOCJP,IORG,AMEAN,AMEM                                        *
```

```
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: DIMEN/LOGFLG                            *
C    ENTCOD/IBATCH                                                       *
C    EVTCOD/IPEREV                                                       *
C    OFFSET/IBATST                                                       *
C    UNITS/ICONST                                                        *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: ENTER,ENTINT,LOG,RDERR,SCHDL                          *
C  Calling Modules: EVENT                                                *
C  Scheduled by: INTLC,PERIOD                                            *
C                                                                        *
C************************************************************************
C
      SUBROUTINE PERIOD
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/ENTCOD/ INTARV,IBATCH,NRLSIN,IPRTEN
      COMMON/EVTCOD/ ICLKEV,INTHEV,IARVEV,IPEREV,ICPDEV,LATEVT,IRESEV,
     +  ISTCEV
      COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
     +  IPORTS
      COMMON/UNITS/ IFACT,ISTUD,IADMN,IOTHR,IBTCH,ICONST
      EQUIVALENCE (ATIME,A(1)),(ILOCAT,A(2)),(IMACH,A(3)),(ICLASS,A(4)),
     +  (ISIZE,A(5)),(ALOCJP,A(7)),(INTBAT,A(9)),(IDIAL,A(11)),
     +  (IORG,A(12)),(AMEAN,A(13)),(AMEM,A(14))
C
C**** ENTER PERIODIC REQUIREMENT INTO NETWORK
      ATIME=TNOW
      IF (LOGFLG) CALL LOG(10,-1)
      IF (ILOCAT .GE. IBATST) THEN
          CALL ENTER(IBATCH,A)
      ELSE
          CALL ENTINT
      ENDIF
C
C**** SCHEDULE NEXT PERIODIC EVENT
      READ(ICONST,END=100,ERR=200) XTIME,ILOCAT,IMACH,ICLASS,ISIZE
      XTIME=XTIME - TNOW
      INTBAT=0
      IDIAL=0
      IORG=ICLASS
      ALOCJP=REAL(IMACH)
      IF (LOGFLG) CALL LOG(11,-1)
      CALL SCHDL(IPEREV,XTIME,A)
  100 RETURN
  200 CALL RDERR(ICONST)
      END
C************************************************************************
C                                                                        *
C  Name: PRTCON                                                          *
C  Module Number: 1.1.2                                                  *
```

```
C  Function: Prints the variable file if ISUM = true at                *
C            initialization.                                            *
C  Attributes Referenced: none                                          *
C  Global (XX) Variables Referenced: IWEEK,IWPCLS,ISTOP,STAINV          *
C  Common Blocks/Variables Used: ANYVAL/ANYDST                          *
C     CPSTAT/IMPL,CNVCPU                                                *
C     DIMEN/NUMCLS,NUMCPU,NUMPRT,NUMLOC,NUMGAT,NUMSIZ                   *
C     FACTOR/DISTRB                                                     *
C     INVALS/DIALUP                                                     *
C     INTPRB/BATINT,ONEJOB,WPPROB                                       *
C     OFFSET/IPORTS                                                     *
C     PRTPRM/PRTSPD,DUPRT,PRTTAB                                        *
C     SCHEDL/SCHCPU,SCHBAT,IPRGAT,ICRGAT                                *
C  Common Blocks/Variables Changed: none                                *
C  Modules Called: none                                                 *
C  Calling Modules: INTLC                                               *
C  Scheduled by: none                                                   *
C                                                                       *
C***********************************************************************
C
      SUBROUTINE PRTCON
      PARAMETER (MAXLOC=10,MAXCPU=6,MAXCLS=6,MAXSIZ=6,MAXPRT=10)
      PARAMETER (MAXGAT=4)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/ANYVAL/ ANYDST(MAXCLS,MAXCPU)
      COMMON/CPSTAT/ ITRMOL,ITRMWP,ITRMAC,IEXJOB,IMEM,ICPU,INOUT,IMPL,
     +  CNVCPU(MAXCPU)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/FACTOR/ DISTRB(MAXCLS,MAXSIZ,6,2)
      COMMON/INVALS/ ARRATE(MAXLOC,24),DIALUP(MAXCPU,24),FRTABL(MAXLOC,
     + MAXCPU,MAXCLS,MAXSIZ)
      COMMON/INTPRB/ ONEJOB,WPPROB(MAXCLS),BATINT(MAXCPU,MAXCLS)
      COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
     +  IPORTS
      COMMON/PRTPRM/ PRTSPD(MAXPRT),DUPRT,PRTTAB(MAXCPU,MAXLOC,MAXPRT),
     +  PCPRTM,PCPRTS
      COMMON/SCHEDL/ SCHBAT(MAXGAT,7,2),SCHCPU(MAXCPU,7,2),
     +  IPRGAT(MAXGAT,MAXPRT),ICRGAT(MAXCPU,MAXGAT)
      EQUIVALENCE (IWEEK,XX(3)),(IWPCLS,XX(4)),(ISTOP,XX(6)),
     +    (ISUM,XX(17)),(STAINV,XX(19))
C
C**** PRINT ALL INPUT VALUES
      WRITE(NPRNT,200)
      DO 10  I=1,NUMCLS
         WRITE(NPRNT,210) I,(ANYDST(I,J), J=1,NUMCPU)
   10 CONTINUE
      WRITE(NPRNT,300)
      DO 15  I=1,NUMCPU
         WRITE(NPRNT,310) I,(BATINT(I,J), J=1,NUMCLS)
   15 CONTINUE
      WRITE(NPRNT,400)
```

```
      DO 20  I=1,NUMCPU
          WRITE(NPRNT,410) I,(DIALUP(I,J), J=1,24)
   20 CONTINUE
      WRITE(NPRNT,500) ONEJOB,(WPPROB(I), I=1,NUMCLS)
      WRITE(NPRNT,600) (PRTSPD(I), I=1,NUMPRT)
      WRITE(NPRNT,650) DUPRT
      DO 25  I=1,NUMCPU
          DO 23  J=1,NUMLOC
              WRITE(NPRNT,660)  I,J,(PRTTAB(I,J,K), K=1,NUMPRT)
   23     CONTINUE
   25 CONTINUE
C
C**** PRINT INITIAL CPU STATES
      WRITE(NPRNT,700)
      DO 27  I=1,NUMCPU
          WRITE(NPRNT,750)  I,XX(I+ICPUOS),SS(I*10-10+IMPL),
     +      XX(I+IPORTS),CNVCPU(I)
   27 CONTINUE
      WRITE(NPRNT,800)
      DO 40  I=1,NUMCLS
          DO 30  J=1,NUMSIZ
              WRITE(NPRNT,810) I,J,((DISTRB(I,J,K,L), L=1,2), K=1,6)
   30     CONTINUE
   40 CONTINUE
C
C**** PRINT COMPUTER SCHEDULES
      WRITE(NPRNT,850)
      DO 50  I=1,NUMCPU
          WRITE(NPRNT,870) I,((SCHCPU(I,J,K), J=1,7), K=1,2)
   50 CONTINUE
C
C**** PRINT BATCH(GATES) SCHEDULES
      WRITE(NPRNT,860)
      DO 60  I=1,NUMGAT
          WRITE(NPRNT,870) I,((SCHBAT(I,J,K), J=1,7), K=1,2)
   60 CONTINUE
C
C**** PRINT PRINTERS CLOSING WITH GATES
      WRITE(NPRNT,900)
      DO 70  I=1,NUMGAT
          WRITE(NPRNT,910) I,(IPRGAT(I,J), J=1,NUMPRT)
   70 CONTINUE
C
C**** PRINT CARD READERS CLOSING WITH COMPUTER DOWN
      WRITE(NPRNT,920)
      DO 80  I=1,NUMCPU
          WRITE(NPRNT,910) I,(ICRGAT(I,J), J=1,NUMGAT)
   80 CONTINUE
      RETURN
C
C**** FORMAT STATEMENTS
  200 FORMAT('0','******* "ANY" COMPUTER PROBABILITIES *******',/,4X,
     +    'CLASS     PROBABILITIES/COMPUTER')
  210 FORMAT(' ',2X,I5,6F10.4)
  300 FORMAT('0','******* "BATCH" MODE PROBABILITIES WHILE INTERACTIVE',
```

```
     +    ' *******',/,3X,'COMPUTER     PROBABILITY')
  310 FORMAT(' ',I6,4X,6F10.4)
  400 FORMAT('0','******* DIALUP PROBABILITIES *******',/,10X,
     +  'COMPUTER    PROB/HOUR')
  410 FORMAT(' ',10X,I4,6X,12F7.4,/,21X,12F7.4)
  500 FORMAT('0','******* WORD PROCESSING PROBABILITIES *******',/,10X,
     +    'WHEN THERE IS ONE JOB - ',F5.3,/,10X,'FOR ALL CLASSES -',
     +    6F6.3)
  600 FORMAT('0','******* PRINTER SPEEDS *******',/,10X,10F10.0)
  650 FORMAT(' ******* PRINTER PROBABILITES *******',/,'  DIALUP',
     +    F8.4,/,' COMPUTER  LOCATION     PROB/PRINTER')
  660 FORMAT(' ',I4,I10,8X,10F6.3)
  700 FORMAT('0','******* INITIAL CPU STATUS *******',/,10X,
     +  'COMPUTER   UP/DOWN    MPL     # PORTS  CNVCPU')
  750 FORMAT(' ',9X,I5,3F10.0,F10.4)
  800 FORMAT('1','******* DISTRIBUTION FACTOR TABLES *******',/,/,15X,
     +    'NEXT JOB SUBMITTAL     K LINES PRINT',9X,'CPU SECONDS',7X,
     +    'NO JOBS  MEMORY SIZE',9X,'I/O SECONDS',/,1X,'CLASS SIZE',
     +  8X,'MEAN    STD',9X, MEAN   STD',9X,'MEAN',8X,'STD',6X,
     +  'MEAN  STD',3X,'MEAN  STD',7X,'MEAN',8X,'STD')
  810 FORMAT(' ',I3,I6,5X,2F8.0,3X,2F8.0,2X,2F11.4,2X,2F5.0,2X,2F5.0,
     +  2X,2F11.4)
  850 FORMAT('1*******  COMPUTER SCHEDULES *******',/,2X,'COMPID',10X,
     +    'UP/DOWN')
  860 FORMAT('0******* BATCH (GATE) SCHEDULES *******',/,'   GATE',10X,
     +    'OPEN/CLOSE')
  870 FORMAT(' ',I5,5X,7F8.0,/,11X,7F8.0)
  900 FORMAT('0','******* PRINTERS CLOSING WITH GATES ******',/,10X,
     +  ' GATE',20X,'PRINTERS')
  910 FORMAT(' ',10X,I3,10X,10I5)
  920 FORMAT('0','******* CARD READERS DOWN WITH COMPUTERS ******',
     +  /,10X,'COMPUTER',10X,'CARD READERS')
      END
C*************************************************************************
C                                                                        *
C  Name: RDERR                                                           *
C  Module Number: 1.C.5                                                  *
C  Function: Prints the unit number when a read error occurs.            *
C  Attributes Referenced: none                                           *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: none                                    *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: none                                                  *
C  Calling Modules: INTLC,PERIOD,UPDARV,UPDFRQ                           *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE RDERR(IUNIT)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
C
      WRITE(NPRNT,10)  IUNIT
```

```
   10 FORMAT(' ','UNIT ',I2,' READ ERROR')
      STOP
      END
C*****************************************************************************
C                                                                             *
C  Name: RDPARM                                                               *
C  Module Number: 1.1.1                                                       *
C  Function: Reads the input variable file at initialization.                 *
C  Attributes Referenced: none                                                *
C  Global (XX) Variables Referenced: IWEEK,IWPCLS,ISTOP,XLOGON,               *
C    XLOGOF,ISUM,INTER                                                        *
C  Common Blocks/Variables Used: ANYVAL/ANYDST                                *
C    CPSTAT/IMPL,CNVCPU                                                       *
C    DIMEN/NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG                   *
C    FACTOR/DISTRB                                                            *
C    INVALS/DIALUP                                                            *
C    INTPRB/ONEJOB,WPPROB,BATINT                                              *
C    LOGPRM/LOGON                                                             *
C    OFFSET/IBATST,ICPUOS,IPORTS                                              *
C    PRTPRM/PRTSPD,DUPRT,PRTTAB,PCPRTM,PCPRTS                                 *
C    SCHEDL/SCHBAT,SCHCPU,IPRGAT,ICRGAT                                       *
C    UNITS/IFACT                                                              *
C    VARCOM/CPULTM,CPULTS,TRMLTM,TRMLTS,OPRNM,OPRSTD,XLOGMN,XLOGST            *
C    WAITCM/TRMWAT,WAITMN,WAITST,PORTWT                                       *
C  Common Blocks/Variables Changed: ANYVAL/ANYDST                             *
C    CPSTAT/IMPL,CNVCPU                                                       *
C    DIMEN/NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG                   *
C    FACTOR/DISTRB                                                            *
C    INVALS/DIALUP                                                            *
C    INTPRB/ONEJOB,WPPROB,BATINT                                              *
C    LOGPRM/LOGON                                                             *
C    OFFSET/IBATST                                                            *
C    PRTPRM/PRTSPD,DUPRT,PRTTAB,PCPRTM,PCPRTS                                 *
C    SCHEDL/SCHBAT,SCHCPU,IPRGAT,ICRGAT                                       *
C    VARCOM/CPULTM,CPULTS,TRMLTM,TRMLTS,OPRNM,OPRSTD,XLOGMN,XLOGST            *
C    WAITCM/TRMWAT,WAITMN,WAITST,PORTWT                                       *
C  Modules Called: none                                                       *
C  Calling Modules: INTLC                                                     *
C  Scheduled by: none                                                         *
C                                                                             *
C*****************************************************************************
C
      SUBROUTINE RDPARM
      PARAMETER (MAXLOC=10,MAXCPU=6,MAXCLS=6,MAXSIZ=6,MAXPRT=10)
      PARAMETER (MAXGAT=4,MAXWAT=5)
      LOGICAL LOGNL,VARNL
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/ANYVAL/ ANYDST(MAXCLS,MAXCPU)
      COMMON/CPSTAT/ ITRMOL,ITRMWP,ITRMAC,IEXJOB,IMEM,ICPU,INOUT,IMPL,
     +  CNVCPU(MAXCPU)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
```

```
      COMMON/FACTOR/ DISTRB(MAXCLS,MAXSIZ,6,2)
      COMMON/INVALS/ ARRATE(MAXLOC,24),DIALUP(MAXCPU,24),FRTABL(MAXLOC,
     + MAXCPU,MAXCLS,MAXSIZ)
      COMMON/INTPRB/ ONEJOB,WPPROB(MAXCLS),BATINT(MAXCPU,MAXCLS)
      COMMON/LOGPRM/ LOGON(25),LOGCNT(25)
      COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
     +  IPORTS
      COMMON/PRTPRM/ PRTSPD(MAXPRT),DUPRT,PRTTAB(MAXCPU,MAXLOC,MAXPRT),
     +  PCPRTM,PCPRTS
      COMMON/SCHEDL/ SCHBAT(MAXGAT,7,2),SCHCPU(MAXCPU,7,2),
     +  IPRGAT(MAXGAT,MAXPRT),ICRGAT(MAXCPU,MAXGAT)
      COMMON/UNITS/ IFACT,ISTUD,IADMN,IOTHR,IBTCH,ICONST
      COMMON/VARCOM/ CPULTM,CPULTS,TRMLTM,TRMLTS,OPRMN,OPRSTD,XLOGMN,
     +  XLOGST
      COMMON/WAITCM/ TRMWAT(MAXLOC,MAXWAT),WAITMN,WAITST,PORTWT(MAXWAT)
      EQUIVALENCE (IWEEK,XX(3)),(IWPCLS,XX(4)),(ISTOP,XX(6)),
     +  (XLOGON,XX(7)),(XLOGOF,XX(8)),(ISUM,XX(17)),(INTER,XX(19))
C
C**** NAMELIST STATEMENTS
      NAMELIST/CARDGAT/ ICRGAT
      NAMELIST/LDIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,
     +  IBATST,IWPCLS
      NAMELIST/CONFIG/ ISUM,INTER,IWEEK,ISTOP,XLOGON,XLOGOF,
     +  LOGNL,VARNL
      NAMELIST/PRTGAT/ IPRGAT
      NAMELIST/LOGFLAG/ LOGON
      NAMELIST/VARTIME/ CPULTM,CPULTS,TRMLTM,TRMLTS,OPRMN,OPRSTD,
     +  XLOGMN,XLOGST,WAITMN,WAITST,PCPRTM,PCPRTS
C
C**** READ IN CARD PARAMETERS
      READ(NCRDR,CONFIG)
      WRITE(NPRNT,100)
      WRITE(NPRNT,CONFIG)
      IF (LOGNL) THEN
          READ(NCRDR,LOGFLAG)
          WRITE(NPRNT,LOGFLAG)
      ENDIF
      IF (VARNL) THEN
          READ(NCRDR,VARTIME)
          WRITE(NPRNT,VARTIME)
      ENDIF
C
      READ(IFACT,LDIMEN)
      LOGFLG=.FALSE.
      IF (XLOGON .LE. 0.1) LOGFLG=.TRUE.
C
C**** GET 'ANY' DISTRIBUTIONS
      READ(IFACT,*)
      DO 10  I=1,NUMCLS
          READ(IFACT,200)  (ANYDST(I,J), J=1,NUMCPU)
   10 CONTINUE
C
C**** GET 'BATCH' PROBABILITIES FOR INTERACTIVE
      READ(IFACT,*)
      DO 15  I=1,NUMCPU
```

```
          READ(IFACT,200)  (BATINT(I,J), J=1,NUMCLS)
   15 CONTINUE
C
C**** GET DIALUP PROBABILITIES
      READ(IFACT,*)
      DO 20  I=1,NUMCPU
          READ(IFACT,200)  (DIALUP(I,J), J=1,24)
   20 CONTINUE
C
C**** GET DISTRIBUTION TABLE
      READ(IFACT,*)
      DO 40  I=1,NUMCLS
          DO 30  J=1,NUMSIZ
              READ(IFACT,300) ((DISTRB(I,J,K,L), L=1,2), K=1,5)
              READ(IFACT,310) (DISTRB(I,J,6,L), L=1,2)
   30     CONTINUE
   40 CONTINUE
C
C**** GET WORD PROCESSING PROBAILITIES
      READ(IFACT,*)
      READ(IFACT,200)  ONEJOB
      READ(IFACT,200)  (WPPROB(I), I=1,NUMCLS)
C
C**** GET PRINTER SPEEDS
      READ(IFACT,*)
      DO 60  I=1,NUMPRT
          READ(IFACT,400)  PRTSPD(I)
   60 CONTINUE
C
C**** GET PRINTER PROBABILTIES
      READ(IFACT,*)
      READ(IFACT,200) DUPRT
      DO 65  I=1,NUMCPU
          DO 63  J=1,NUMLOC
              READ(IFACT,200) (PRTTAB(I,J,K), K=1,NUMPRT)
   63     CONTINUE
   65 CONTINUE
C
C**** GET CPU UP/DOWN, MULTIPROGRAM LEVELS, # PORTS, CNVCPU
      READ(IFACT,*)
      DO 70  I=1,NUMCPU
          READ(IFACT,600)  XX(I+ICPUOS),SS(I*10-10+IMPL),XX(I+IPORTS),
     +       CNVCPU(I)
   70 CONTINUE
C
C**** READ IN COMPUTER SCHEDULES
      READ(IFACT,*)
   80 READ(IFACT,500) ICPU
      IF (ICPU .GT. 0) THEN
          READ(IFACT,*) ((SCHCPU(ICPU,J,K), J=1,7), K=1,2)
          GO TO 80
      ENDIF
C
C**** READ IN BATCH (GATE) SCHEDULES
      READ(IFACT,*)
```

```
   90 READ(IFACT,500) IBAT
      IF (IBAT .GT. 0) THEN
          READ(IFACT,*) ((SCHBAT(IBAT,J,K), J=1,7), K=1,2)
          GO TO 90
      ENDIF
C
C**** READ IN WAIT PROBABILITIES
      READ(IFACT,*)
      DO 95  I=1,IBATST-1
          READ(IFACT,200) (TRMWAT(I,J), J=1,MAXWAT)
   95 CONTINUE
C
C**** READ PRINTERS AND CARD READERS ASSOCIATION
      READ(IFACT,*)
      READ(IFACT,PRTGAT)
      READ(IFACT,*)
      READ(IFACT,CARDGAT)
      RETURN
C
C**** FORMAT STATEMENTS
  100 FORMAT('1',50X,'****** INPUT PARAMETERS ******')
  200 FORMAT(8X,12F6.4)
  300 FORMAT(8X,4F5.0,2F10.4,4F5.0)
  310 FORMAT(8X,2F10.4)
  400 FORMAT(8X,F8.0)
  500 FORMAT(8X,I8)
  600 FORMAT(8X,3F8.0,F8.4)
      END
C*************************************************************************
C                                                                        *
C  Name: RESCHD                                                          *
C  Module Number: 1.C.2.1                                                *
C  Function: Reschedules an arrival when computer not available.         *
C  Attributes Referenced: none                                           *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: DIMEN/LOGFLG                            *
C    EVTCOD/LATEVT                                                       *
C    VARCOM/CPULTM,CPULTS                                                *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: LOG,SCHDL                                             *
C  Calling Modules: ENTINT                                               *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE RESCHD
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/EVTCOD/ ICLKEV,INTHEV,IARVEV,IPEREV,ICPDEV,LATEVT,IRESEV,
     +  ISTCEV
      COMMON/VARCOM/ CPULTM,CPULTS,TRMLTM,TRMLTS,OPRMN,OPRSTD,XLOGMN,
```

```
   90 READ(IFACT,500) IBAT
      IF (IBAT .GT. 0) THEN
          READ(IFACT,*) ((SCHBAT(IBAT,J,K), J=1,7), K=1,2)
          GO TO 90
      ENDIF
C
C**** READ IN WAIT PROBABILITIES
      READ(IFACT,*)
      DO 95  I=1,IBATST-1
          READ(IFACT,200) (TRMWAT(I,J), J=1,MAXWAT)
   95 CONTINUE
C
C**** READ PRINTERS AND CARD READERS ASSOCIATION
      READ(IFACT,*)
      READ(IFACT,PRTGAT)
      READ(IFACT,*)
      READ(IFACT,CARDGAT)
      RETURN
C
C**** FORMAT STATEMENTS
  100 FORMAT('1',50X,'****** INPUT PARAMETERS ******')
  200 FORMAT(8X,12F6.4)
  300 FORMAT(8X,4F5.0,2F10.4,4F5.0)
  310 FORMAT(8X,2F10.4)
  400 FORMAT(8X,F3.0)
  500 FORMAT(8X,I8)
  600 FORMAT(8X,3F8.0,F8.4)
      END
C*************************************************************************
C                                                                        *
C  Name: RESCHD                                                          *
C  Module Number: 1.C.2.1                                                *
C  Function: Reschedules an arrival when computer not available.         *
C  Attributes Referenced: none                                           *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: DIMEN/LOGFLG                            *
C    EVTCOD/LATEVT                                                       *
C    VARCOM/CPULTM,CPULTS                                                *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: LOG,SCHDL                                             *
C  Calling Modules: ENTINT                                               *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE RESCHD
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/EVTCOD/ ICLKEV,INTHEV,IARVEV,IPEREV,ICPDEV,LATEVT,IRESEV,
     +   ISTCEV
      COMMON/VARCOM/ CPULTM,CPULTS,TRMLTM,TRMLTS,OPRMN,OPRSTD,XLOGMN,
```

```
     +  XLOGST
C
C**** COMPUTER NOT AVAILABLE-SCHEDULE LATER ARRIVAL
C
      XTIME=RNORM(CPULTM,CPULTS,1)
      IF (LOGFLG) CALL LOG(12,-1)
      CALL SCHDL(LATEVT,XTIME,A)
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: RESORC                                                          *
C  Module Number: 1.E.3                                                  *
C  Function: Opens/closes card readers, printers, and computers.         *
C            If computer closes then calls STPINT to take inter-         *
C            users off.                                                  *
C  Attributes Referenced: ICOMP,IGATE,ITYPE                              *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: DIMEN/NUMGAT,NUMPRT,LOGFLG              *
C    OFFSET/ICPUOS,IGATOS,IPRTOS                                         *
C    SCHEDL/ICRGAT,IPRGAT                                                *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: ALTER,CLOSX,LOG,OPEN,STPINT                           *
C  Calling Modules: EVENT                                                *
C  Scheduled by: CALSCH                                                  *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE RESORC
      PARAMETER (MAXGAT=4,MAXCPU=6,MAXPRT=10)
C
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
     +  IPORTS
      COMMON/SCHEDL/ SCHBAT(MAXGAT,7,2),SCHCPU(MAXCPU,7,2),
     +  IPRGAT(MAXGAT,MAXPRT),ICRGAT(MAXCPU,MAXGAT)
      EQUIVALENCE (ICOMP,IGATE,A(2)),(ITYPE,A(3))
C
      IF (ICOMP .GE. 10) THEN
C         **** COMPUTER CHANGE ****
          XX(ICOMP)=REAL(ITYPE)
          IF (ITYPE .EQ. 0) THEN
              NCOMP=ICOMP
              CALL STPINT(ICOMP)
C             **** CLOSE COMPUTER'S CARD READERS ****
              DO 50  I=1,NUMGAT
                  IF (ICRGAT(NCOMP-ICPUOS,I).EQ.1) CALL CLOSX(I+IGATOS)
   50         CONTINUE
          ELSE
C             **** OPEN COMPUTERS CARD READERS ****
              DO 75  I=1,NUMGAT
                  IF (ICRGAT(ICOMP-ICPUOS,I).EQ.1) CALL OPEN(I+IGATOS)
```

```
   75         CONTINUE
            ENDIF
         ELSE
C           **** BATCH GATE CHANGE ****
            IF(ITYPE .EQ. 0) THEN
               CALL CLOSX(IGATE)
C              **** CLOSE THE PRINTERS ****
               DO 100  I=1,NUMPRT
                  IF (IPRGAT(IGATE,I) .EQ. 1) CALL ALTER(IPRTOS+I,-1)
  100          CONTINUE
            ELSE
               CALL OPEN(IGATE)
C              **** OPEN THE PRINTERS ****
               DO 200  I=1,NUMPRT
                  IF (IPRGAT(IGATE,I) .EQ. 1) CALL ALTER(IPRTOS+I,+1)
  200          CONTINUE
            ENDIF
         ENDIF
         IF (LOGFLG) CALL LOG(18,-1)
         RETURN
         END
C*****************************************************************************
C                                                                            *
C  Name: RTPRT                                                               *
C  Module Number: 1.E.8.1                                                    *
C  Function: Determines print time and printer then enters into              *
C            network.                                                        *
C  Attributes Referenced: APRTL,INTBAT,IDIAL,AJUMP                           *
C  Global (XX) Variables Referenced: none                                    *
C  Common Blocks/Variables Used: DIMEN/NUMPRT,LOGFLG                         *
C    ENTCOD/IPRTEN                                                           *
C    EVTCOD/INTHEV                                                           *
C    OFFSET/ICPUOS                                                           *
C    PRTPRM/DUPRT,PCPRTM,PCPRTS,PRTTAB,PRTSPD                                *
C  Common Blocks/Variables Changed: none                                     *
C  Modules Called: ENTER,LOG,SCHDL                                           *
C  Calling Modules: CPUDPT                                                   *
C  Scheduled by: none                                                        *
C                                                                            *
C*****************************************************************************
C
      SUBROUTINE RTPRT(IMACH,LOCAT)
      PARAMETER (MAXCPU=6,MAXLOC=10,MAXPRT=10)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/ENTCOD/ INTARV,IBATCH,NRLSIN,IPRTEN
      COMMON/EVTCOD/ ICLKEV,INTHEV,IARVEV,IPEREV,ICPDEV,LATEVT,IRESEV,
     +  ISTCEV
      COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
     +  IPORTS
      COMMON/PRTPRM/ PRTSPD(MAXPRT),DUPRT,PRTTAB(MAXCPU,MAXLOC,MAXPRT),
```

```
     +  PCPRTM,PCPRTS
      EQUIVALENCE (APRTL,A(8)),(INTBAT,A(9)),(IDIAL,A(11)),(AJUMP,A(15))
C
C**** ROUTE TO PRINTER
      RN=DRAND(1)
      AJUMP=1.
      IF ((IDIAL .EQ. 1).AND.(INTBAT .EQ. 1).AND.(RN .LE. DUPRT)) THEN
          AJUMP=AJUMP+REAL(NUMPRT)
          APRTL=(APRTL*1000)/RNORM(PCPRTM,PCPRTS,1)*60.
      ELSE
          IF ((INTBAT.EQ.1).AND.(XX(IMACH+ICPUOS).GT.0.5))
     +       CALL SCHDL(INTHEV,1.0,A)
          DO 50  I=1,NUMPRT-1
              IF (RN .GE. PRTTAB(IMACH,LOCAT,I)) AJUMP=AJUMP+1.
   50     CONTINUE
C         **** DETERMINE PRINTER TIME ****
          II=NINT(AJUMP)
          APRTL=(APRTL*1000./PRTSPD(II))*60.
      ENDIF
      CALL ENTER(IPRTEN,A)
C
      IF (LOGFLG) CALL LOG(13,-1)
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: SESOVR                                                          *
C  Module Number: 1.C.6                                                  *
C  Function: Decrements CPU states when interactive session over.        *
C            Enters into network release.  If another waiting for        *
C            port then schedules arrival.                                *
C  Attributes Referenced: ILOCAT,IMACH,IDIAL,IORG,AJUMP                  *
C  Global (XX) Variables Referenced: IWPCLS                              *
C  Common Blocks/Variables Used: CPSTAT/ITRMOL,ITRMWP,ITRMAC             *
C    DIMEN/LOGFLG                                                        *
C    ENTCOD/NRLSIN                                                       *
C    EVTCOD/ISTCEV                                                       *
C    OFFSET/IPORTS                                                       *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: ENTER,LOG,RMOVE,SCHDL                                 *
C  Calling Modules: INTHND,STPINT                                        *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE SESOVR
      PARAMETER (MAXCPU=6)
      DIMENSION ATEMP(100)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/CPSTAT/ ITRMOL,ITRMWP,ITRMAC,IEXJOB,IMEM,ICPU,INOUT,IMPL,
     +  CNVCPU(MAXCPU)
      LOGICAL LOGFLG
```

```
     +  PCPRTM,PCPRTS
      EQUIVALENCE (APRTL,A(8)),(INTBAT,A(9)),(IDIAL,A(11)),(AJUMP,A(15))
C
C**** ROUTE TO PRINTER
      RN=DRAND(1)
      AJUMP=1.
      IF ((IDIAL .EQ. 1).AND.(INTBAT .EQ. 1).AND.(RN .LE. DUPRT)) THEN
          AJUMP=AJUMP+REAL(NUMPRT)
          APRTL=(APRTL*1000)/RNORM(PCPRTM,PCPRTS,1)*60.
      ELSE
          IF ((INTBAT.EQ.1).AND.(XX(IMACH+ICPUOS).GT.0.5))
     +       CALL SCHDL(INTHEV,1.0,A)
          DO 50  I=1,NUMPRT-1
              IF (RN .GE. PRTTAB(IMACH,LOCAT,I)) AJUMP=AJUMP+1.
   50     CONTINUE
C         **** DETERMINE PRINTER TIME ****
          II=NINT(AJUMP)
          APRTL=(APRTL*1000./PRTSPD(II))*60.
      ENDIF
      CALL ENTER(IPRTEN,A)
C
      IF (LOGFLG) CALL LOG(13,-1)
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: SESOVR                                                          *
C  Module Number: 1.C.6                                                  *
C  Function: Decrements CPU states when interactive session over.        *
C            Enters into network release.  If another waiting for        *
C            port then schedules arrival.                                *
C  Attributes Referenced: ILOCAT,IMACH,IDIAL,IORG,AJUMP                  *
C  Global (XX) Variables Referenced: IWPCLS                              *
C  Common Blocks/Variables Used: CPSTAT/ITRMOL,ITRMWP,ITRMAC             *
C    DIMEN/LOGFLG                                                        *
C    ENTCOD/NRLSIN                                                       *
C    EVTCOD/ISTCEV                                                       *
C    OFFSET/IPORTS                                                       *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: ENTER,LOG,RMOVE,SCHDL                                 *
C  Calling Modules: INTHND,STPINT                                        *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE SESOVR
      PARAMETER (MAXCPU=6)
      DIMENSION ATEMP(100)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/CPSTAT/ ITRMOL,ITRMWP,ITRMAC,IEXJOB,IMEM,ICPU,INOUT,IMPL,
     +  CNVCPU(MAXCPU)
      LOGICAL LOGFLG
```

```
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/ENTCOD/ INTARV,IBATCH,NRLSIN,IPRTEN
      COMMON/EVTCOD/ ICLKEV,INTHEV,IARVEV,IEREV,ICPDEV,LATEVT,IRESEV,
     +  ISTCEV
      COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
     +  IPORTS
      EQUIVALENCE (ILOCAT,A(2)),(IMACH,A(3)),(ALOCAT,A(7)),
     +  (IDIAL,A(11)),(IORG,A(12))
      EQUIVALENCE (IWPCLS,XX(4))
C
C**** DECREMENT CPU STATES
      INDEX=IMACH*10-10
      SS(INDEX+ITRMOL)=SS(INDEX+ITRMOL)-1.
      IF (IORG .EQ. IWPCLS)  SS(INDEX+ITRMWP)=SS(INDEX+ITRMWP)-1.
      SS(INDEX+ITRMAC)=SS(INDEX+ITRMAC)-1.
      XX(IPORTS+IMACH)=XX(IPORTS+IMACH)+1.
C
C**** IS A USER WAITING FOR A PORT?
      IF (NNQ(IPORTS+IMACH) .GT. 0) THEN
C         **** YES - SCHEDULE TO LOGON ****
          CALL RMOVE(1,IPORTS+IMACH,ATEMP)
          CALL SCHDL(ISTCEV,0.1,ATEMP)
      ENDIF
C
C**** RELEASE RESOURCE
      IF (IDIAL .EQ. 1) THEN
          ALOCAT=0.0
      ELSE
          ALOCAT=REAL(ILOCAT)
      ENDIF
      CALL ENTER(NRLSIN,A)
C
      IF (LOGFLG) CALL LOG(7,INDEX)
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: SETCPU                                                          *
C  Module Number: 1.E.4                                                  *
C  Function: Sets the CPU's states and session time for interactive      *
C            arrival.                                                    *
C  Attributes Referenced: IMACH,ICLASS,ISIZE,IJOBS,AMEAN                 *
C  Global (XX) Variables Referenced: IWPCLS                              *
C  Common Blocks/Variables Used: CPSTAT/ITRMOL,ITRMAC,ITRMWP             *
C    DIMEN/LOGFLG                                                        *
C    EVTCOD/INTHEV                                                       *
C    FACTOR/DISTRB                                                       *
C    OFFSET/IPORTS                                                       *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: LOG,SCHDL                                             *
C  Calling Modules: EVENT,LOGON                                          *
C  Scheduled by: SESOVR                                                  *
C                                                                        *
C*************************************************************************
C
```

```
      SUBROUTINE SETCPU
      PARAMETER (MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/CPSTAT/ ITRMOL,ITRMWP,ITRMAC,IEXJOB,IMEM,ICPU,INOUT,IMPL,
     +  CNVCPU(MAXCPU)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/EVTCOD/ ICLKEV,INTHEV,IARVEV,IPEREV,ICPDEV,LATEVT,IRESEV,
     +  ISTCEV
      COMMON/FACTOR/ DISTRB(MAXCLS,MAXSIZ,6,2)
      COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
     +  IPORTS
      EQUIVALENCE (IMACH,A(3)),(ICLASS,A(4)),(ISIZE,A(5)),
     +  (IJOBS,A(10)),(AMEAN,A(13))
      EQUIVALENCE (IWPCLS,XX(4))
C
C**** DECREMENT PORT AVAILABLE
      XX(IPORTS+IMACH)=XX(IPORTS+IMACH)-1.
C
C**** UPDATE CPU STATES
      INDEX=IMACH*10-10
      SS(INDEX+ITRMOL)=SS(INDEX+ITRMOL)+1.
      SS(INDEX+ITRMAC)=SS(INDEX+ITRMAC)+1.
      IF (ICLASS .EQ. IWPCLS) SS(INDEX+ITRMWP)=SS(INDEX+ITRMWP)+1.
C
C**** GET NUMBER OF JOBS THIS SESSION
      XMEAN=DISTRB(ICLASS,ISIZE,4,1)
      STD=DISTRB(ICLASS,ISIZE,4,2)
      IJOBS=NINT(RNORM(XMEAN,STD,1))
C
C**** GET JOBS SUBMITTAL DISTRIBUTION
      AMEAN=DISTRB(ICLASS,ISIZE,1,1)
C
C**** ENTER TO INTERACTIVE HANDLER
      IF (LOGFLG) CALL LOG(20,INDEX)
      CALL SCHDL(INTHEV,EXPON(AMEAN,1),A)
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: SETINT                                                          *
C  Module Number: 1.C.7                                                  *
C  Function: Sets interactive jobs print and 'mode' characteristics.     *
C  Attributes Referenced: IMACH,ICLASS,INTBAT,IJOBS                      *
C  Global (XX) Variables Referenced: IWPCLS                              *
C  Common Blocks/Variables Used: DIMEN/LOGFLG                            *
C    INTPRB/ONEJOB,WPPROB,BATINT                                         *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: LOG                                                   *
C  Calling Modules: INTHND                                               *
C  Scheduled by: none                                                    *
C                                                                        *
```

```
C*************************************************************************
C
      SUBROUTINE SETINT
C
      PARAMETER (MAXCPU=6,MAXCLS=6)
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/INTPRB/  ONEJOB,WPPROB(MAXCLS),BATINT(MAXCPU,MAXCLS)
      EQUIVALENCE (IMACH,A(3)),(ICLASS,A(4)),(INTBAT,A(9)),(IJOBS,A(10))
      EQUIVALENCE (IWPCLS,XX(4))
C
C**** SHOULD WE CHANGE TO WORD PROCESSING (PRINT)
      IF (ICLASS .NE. IWPCLS) THEN
          RN=DRAND(1)
          IF (IJOBS .LE. 0) THEN
              IF (RN .LE. ONEJOB)  ICLASS=IWPCLS
          ELSE
              IF (RN .LE. WPPROB(ICLASS)) ICLASS=IWPCLS
          ENDIF
      ENDIF
C
C**** IS THE USER SUBMITTING THIS JOB AS WAIT OR BATCH MODE
      RN=DRAND(1)
      IF (RN .LE. BATINT(IMACH,ICLASS)) THEN
          INTBAT=0
      ELSE
          INTBAT=1
      ENDIF
      IF (LOGFLG) CALL LOG(14,-1)
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: STPINT                                                          *
C  Module Number: 1.E.3.1                                                *
C  Function: Called when computer goes down to release interactive       *
C            users.                                                      *
C  Attributes Referenced: IMACH,ICODE                                    *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: DIMEN/LOGFLG                            *
C    EVTCOD/INTHEV                                                       *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: COPY,RMOVE,SESOVR                                     *
C  Calling Modules: RESORC                                               *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      SUBROUTINE STPINT(IDOWN)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
```

```
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/EVTCOD/ ICLKEV,INTHEV,IARVEV,IPEREV,ICPDEV,LATEVT,IRESEV,
     +  ISTCEV
      COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
     +  IPORTS
      EQUIVALENCE (IMACH,A(3)),(ICODE,A(16))
C
C**** COMPUTER IS DOWN - BLOW THOSE USERS OFF
      IEND=NNQ(NCLNR)
      IF (IEND .GT. 0) THEN
          ISPOT=1
   10     CALL COPY(ISPOT,NCLNR,A)
          IF ((IMACH.EQ.IDOWN).AND.(ICODE.EQ.INTHEV)) THEN
              CALL RMOVE(ISPOT,NCLNR,A)
              CALL SESOVR
              IEND=IEND-1
          ELSE
              ISPOT=ISPOT+1
          ENDIF
          IF (ISPOT .LE. IEND) GO TO 10
      ENDIF
C
C**** REMOVE ANY USERS WAITING FOR A PORT
      IQ=IPORTS+IDOWN
      IEND=NNQ(IQ)
      IF (IEND .GT. 0) THEN
          ISPOT=1
   20     CALL COPY(ISPOT,NCLNR,A)
          IF (IMACH.EQ.IDOWN) THEN
              CALL RMOVE(ISPOT,NCLNR,A)
              CALL SESOVR
              IEND=IEND-1
          ELSE
              ISPOT=ISPOT+1
          ENDIF
          IF (ISPOT .LE. IEND) GO TO 20
      ENDIF
      IF (LOGFLG) CALL LOG(21,-1)
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: UPDARV                                                          *
C  Module Number: 1.C.8                                                  *
C  Function: Updates the arrival rates for all locations.                *
C  Attributes Referenced: none                                           *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: DIMEN/LOGFLG                            *
C    INVALS/ARRATE                                                       *
C    UNITS/ISTUD,IADMN,IOTHR,IBTCH                                       *
C  Common Blocks/Variables Changed: INVALS/ARRATE                        *
C  Modules Called: RDERR                                                 *
C  Calling Modules: DYPROC,INTLC                                         *
```

```
C  Scheduled by: none                                                    *
C                                                                        *
C************************************************************************
C
      SUBROUTINE UPDARV
      PARAMETER (MAXLOC=10,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/INVALS/  ARRATE(MAXLOC,24),DIALUP(MAXCPU,24),
     +     FRTABL(MAXLOC,MAXCPU,MAXCLS,MAXSIZ)
      COMMON/UNITS/ IFACT,ISTUD,IADMN,IOTHR,IBTCH,ICONST
C
C**** UDATE ARRIVAL RATES
      IUNIT=ISTUD
      DO 100 I=1,3
          READ(ISTUD,ERR=500) (ARRATE(I,J), J=1,24)
  100 CONTINUE
      IUNIT=IADMN
      DO 200 I=4,6
          READ(IADMN,ERR=500) (ARRATE(I,J), J=1,24)
  200 CONTINUE
      IUNIT=IOTHR
      DO 300 I=7,7
          READ(IOTHR,ERR=500) (ARRATE(I,J), J=1,24)
  300 CONTINUE
      IUNIT=IBTCH
      DO 400 I=8,10
          READ(IBTCH,ERR=500) (ARRATE(I,J), J=1,24)
  400 CONTINUE
      IF (LOGFLG) THEN
          WRITE(NPRNT,700)
          DO 450 I=1,10
              WRITE(NPRNT,710) I,(ARRATE(I,J), J=1,12)
              WRITE(NPRNT,720) (ARRATE(I,J), J=13,24)
  450     CONTINUE
      ENDIF
      RETURN
  500 CALL RDERR(IUNIT)
  700 FORMAT('0  ARRIVAL RATES')
  710 FORMAT(' ',I5,12F8.2)
  720 FORMAT(' ',5X,12F8.2)
      END
C************************************************************************
C                                                                        *
C  Name: UPDFRQ                                                          *
C  Module Number: 1.C.9                                                  *
C  Function: Updates the cummulative frequency distributions.            *
C  Attributes Referenced: none                                           *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: DIMEN/NUMCLS,NUMCPU,NUMSIZ              *
C    INVALS/FRTABL                                                       *
C    UNITS/ISTUD,IADMN,IOTHR,IBTCH                                       *
```

```
C  Common Blocks/Variables Changed: INVALS/FRTABL                          *
C  Modules Called: RDERR                                                   *
C  Calling Modules: INTLC,WKPROC                                           *
C  Scheduled by: none                                                      *
C                                                                          *
C**************************************************************************
C
      SUBROUTINE UPDFRQ
      PARAMETER (MAXLOC=10,MAXCPU=6,MAXCLS=6,MAXSIZ=6)
C**** COMMON BLOCKS
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/INVALS/  ARRATE(MAXLOC,24),DIALUP(MAXCPU,24),
     +    FRTABL(MAXLOC,MAXCPU,MAXCLS,MAXSIZ)
      COMMON/UNITS/ IFACT,ISTUD,IADMN,IOTHR,IBTCH,ICONST
C
C**** UPDATE FREQUENCY TABLES
      IUNIT=ISTUD
      DO 100  I=1,3
          READ(ISTUD,ERR=500)  (((FRTABL(I,J,K,L), L=1,NUMSIZ),
     +      K=1,NUMCLS), J=1,NUMCPU+1)
  100 CONTINUE
      IUNIT=IADMN
      DO 200  I=4,6
          READ(IADMN,ERR=500)  (((FRTABL(I,J,K,L), L=1,NUMSIZ),
     +      K=1,NUMCLS), J=1,NUMCPU+1)
  200 CONTINUE
      IUNIT=IOTHR
      DO 300  I=7,7
          READ(IOTHR,ERR=500)  (((FRTABL(I,J,K,L), L=1,NUMSIZ),
     +      K=1,NUMCLS), J=1,NUMCPU+1)
  300 CONTINUE
      IUNIT=IBTCH
      DO 400  I=8,10
          READ(IBTCH,ERR=500)  (((FRTABL(I,J,K,L), L=1,NUMSIZ),
     +      K=1,NUMCLS), J=1,NUMCPU+1)
  400 CONTINUE
C
      RETURN
  500 CALL RDERR(IUNIT)
      END
C**************************************************************************
C                                                                          *
C  Name: WKPROC                                                            *
C  Module Number: 1.E.1.3                                                  *
C  Function: Called at end of week to update arrival rates and             *
C            frequency distributions.                                      *
C  Attributes Referenced: none                                             *
C  Global (XX) Variables Referenced: none                                  *
C  Common Blocks/Variables Used: none                                      *
C  Common Blocks/Variables Changed: none                                   *
C  Modules Called: UPDARV,UPDFRQ                                           *
C  Calling Modules: CLOCK                                                  *
C  Scheduled by: none                                                      *
C                                                                          *
```

```
C*************************************************************************
C
      SUBROUTINE WKPROC
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
C
C**** UPDATE FREQUENCY TABLE
      CALL UPDFRQ
      CALL UPDARV
      CALL CALSCH
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: ARVTIM                                                          *
C  Module Number: 1.F.1                                                  *
C  Function: Returns next arrival time using RATE.                       *
C  Attributes Referenced: none                                           *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: none                                    *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: none                                                  *
C  Calling Modules: ARVL,INTLC                                           *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      FUNCTION ARVTIM(RATE)
C
C**** GIVEN A ARRIVAL RATE - WHEN IS THEN NEXT ARRIVAL
      XMEAN=60./RATE*60.
      ARVTIM=EXPON(XMEAN,1)
      IF (ARVTIM .LE. 0.0) ARVTIM=0.01
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: CPUSEC                                                          *
C  Module Number: 1.F.2                                                  *
C  Function: Given a class and size, returns CPU seconds required.       *
C  Attributes Referenced: none                                           *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: FACTOR/DISTRB                           *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: none                                                  *
C  Calling Modules: CPUARV                                               *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      FUNCTION CPUSEC(ICLASS,ISIZE)
      PARAMETER (MAXCLS=6,MAXSIZ=6)
C
```

```
C**** COMMON BLOCKS
      COMMON/FACTOR/ DISTRB(MAXCLS,MAXSIZ,6,2)
C
C**** WHAT IS THE CPU TIME REQUIRED
      XMEAN=DISTRB(ICLASS,ISIZE,3,1)
      CPUSEC=EXPON(XMEAN,1)
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: FIGRUN                                                          *
C  Module Number: 1.F.3                                                  *
C  Function: Returns a job's run time as a function of computer          *
C            state and the job characteristics.                          *
C            ****** not implemented *****                                *
C  Attributes Referenced: none                                           *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: none                                    *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: none                                                  *
C  Calling Modules: CPUARV,CPUDPT                                        *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      FUNCTION FIGRUN(IMACH,ICLASS,CPUSEC,MEM)
C
C**** FIGURE RUN TIME AS FUNCTION OF CLASS,SIZE AND CPU STATES
      FIGRUN=1000.0
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: MEMSIZ                                                          *
C  Module Number: 1.F.4                                                  *
C  Function: Returns memory size as a function of class and size.        *
C  Attributes Referenced: none                                           *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: FACTOR/DISTRB                           *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: none                                                  *
C  Calling Modules: CPUARV                                               *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      FUNCTION MEMSIZ(ICLASS,ISIZE)
      REAL MEMSIZ
      PARAMETER (MAXCLS=6,MAXSIZ=6)
C
C**** COMMON BLOCKS
      COMMON/FACTOR/ DISTRB(MAXCLS,MAXSIZ,6,2)
C
C**** WHAT IS THE MEMORY REQUIRED
      XMEAN=DISTRB(ICLASS,ISIZE,5,1)
```

```
      STD=DISTRB(ICLASS,ISIZE,5,2)
      MEMSIZ=RNORM(XMEAN,STD,1)
      IF (MEMSIZ .LE. 0.0) MEMSIZ=100.
      RETURN
      END
C*****************************************************************************
C                                                                             *
C  Name: PRTLIN                                                               *
C  Module Number: 1.F.5                                                       *
C  Function: Returns number of print lines as a function of class             *
C            and size.                                                        *
C  Attributes Referenced: none                                                *
C  Global (XX) Variables Referenced: none                                     *
C  Common Blocks/Variables Used: FACTOR/DISTRB                                *
C  Common Blocks/Variables Changed: none                                      *
C  Modules Called: none                                                       *
C  Calling Modules: CPUDPT                                                    *
C  Scheduled by: none                                                         *
C                                                                             *
C*****************************************************************************
C
      FUNCTION PRTLIN(ICLASS,ISIZE)
      PARAMETER (MAXCLS=6,MAXSIZ=6)
C
C**** COMMON BLOCKS
      COMMON/FACTOR/ DISTRB(MAXCLS,MAXSIZ,6,2)
C
C**** HOW MANY K PRINT LINES THIS JOB
      XMEAN=DISTRB(ICLASS,ISIZE,2,1)
      STD=DISTRB(ICLASS,ISIZE,2,2)
      PRTLIN=RNORM(XMEAN,STD,1)
      RETURN
      END
C*****************************************************************************
C                                                                             *
C  Name: USERF                                                                *
C  Module Number: 1.F.6                                                       *
C  Function: Returns log on time or batch submit time.                        *
C  Attributes Referenced: none                                                *
C  Global (XX) Variables Referenced: none                                     *
C  Common Blocks/Variables Used: VARCOM/XLOGMN,XLOGST,OPRMN,OPRSTD            *
C  Common Blocks/Variables Changed: none                                      *
C  Modules Called: none                                                       *
C  Calling Modules: NETWORK                                                   *
C  Scheduled by: none                                                         *
C                                                                             *
C*****************************************************************************
C
      FUNCTION USERF(IFN)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/VARCOM/ CPULTM,CPULTS,TRMLTM,TRMLTS,OPRMN,OPRSTD,XLOGMN,
     +  XLOGST
```

```
C
      GO TO (10,20) IFN
      CALL LOGERR(1)
      USERF=10.0
      RETURN
C**** LOG ON TIME
   10 USERF=RNORM(XLOGMN,XLOGST,1)
      RETURN
C**** OPERATOR BATCH SUBMIT TIME
   20 USERF=RNORM(OPRMN,OPRSTD,1)
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: WAITPO                                                          *
C  Module Number: 1.F.7                                                  *
C  Function: Determines if interactive arrival will wait for a port.     *
C            If won't wait then reschedules arrival.                     *
C  Attributes Referenced: ILOCAT,IMACH                                   *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: DIMEN/LOGFLG                            *
C    EVTCOD/LATEVT                                                       *
C    OFFSET/IPORTS                                                       *
C    WAITCM/PORTWT,WAITMN,WAITST                                         *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: LOG,SCHDL                                             *
C  Calling Modules: LOGON                                                *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
      FUNCTION WAITPO(DUMMY)
      PARAMETER (MAXLOC=10,MAXWAT=5)
C
C**** COMMON BLOCKS
      COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
     + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      LOGICAL LOGFLG
      COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
      COMMON/EVTCOD/ ICLKEV,INTHEV,IARVEV,IPEREV,ICPDEV,LATEVT,IRESEV,
     +  ISTCEV
      COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
     +  IPORTS
      COMMON/WAITCM/ TRMWAT(MAXLOC,MAXWAT),WAITMN,WAITST,PORTWT(MAXWAT)
      EQUIVALENCE (ILOCAT,A(2)),(IMACH,A(3))
C
C**** WILL THE USER WAIT FOR A PORT?
      RN=DRAND(1)
      II=NNC(IPORTS+IMACH)+1
      IF (II .GT. MAXWAT) II=MAXWAT
      IF (RN .LE. PORTWT(II)) THEN
C         **** WILL WAIT ****
          WAITPO=1.0
          IF (LOGFLG) CALL LOG(22,-1)
      ELSE
```

```
C          **** WON'T WAIT -SCHEDULE LATER ARRIVAL ****
           WAITPO=0.0
           XTIME=RNORM(WAITMN,WAITST,1)
           CALL SCHDL(LATEVT,XTIME,A)
           IF (LOGFLG) CALL LOG(23,-1)
       ENDIF
C
       RETURN
       END

C*************************************************************************
C                                                                        *
C  Name: WAITTR                                                          *
C  Module Number: 1.F.8                                                  *
C  Function: Determines if interactive arrival will wait for a           *
C            terminal.  If won't wait then reschedules arrival.          *
C  Attributes Referenced: ILOCAT                                         *
C  Global (XX) Variables Referenced: none                                *
C  Common Blocks/Variables Used: DIMEN/LOGFLG                            *
C    EVTCOD/LATEVT                                                       *
C    OFFSET/ITRMOS                                                       *
C    VARCOM/TRMLTM,TRMLTS                                                *
C    WAITCM/TRMWAT                                                       *
C  Common Blocks/Variables Changed: none                                 *
C  Modules Called: LOG,SHCDL                                             *
C  Calling Modules: ENTINT                                               *
C  Scheduled by: none                                                    *
C                                                                        *
C*************************************************************************
C
       FUNCTION WAITTR(DUMMY)
       PARAMETER (MAXLOC=10,MAXWAT=5)
C
C**** COMMON BLOCKS
       COMMON/SCOM1/ A(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR,
      + NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
       LOGICAL LOGFLG
       COMMON/DIMEN/ NUMLOC,NUMCPU,NUMCLS,NUMSIZ,NUMPRT,NUMGAT,LOGFLG
       COMMON/EVTCOD/ ICLKEV,INTHEV,IARVEV,IPEREV,ICPDEV,LATEVT,IRESEV,
      +  ISTCEV
       COMMON/OFFSET/ ICPUOS,IHLDOS,IRTSTR,ITRMOS,IBATST,IGATOS,IPRTOS,
      +  IPORTS
       COMMON/VARCOM/ CPULTM,CPULTS,TRMLTM,TRMLTS,OPRMN,OPRSTD,XLOGMN,
      +  XLOGST
       COMMON/WAITCM/ TRMWAT(MAXLOC,MAXWAT),WAITMN,WAITST,PORTWT(MAXWAT)
       EQUIVALENCE (ILOCAT,A(2))
C
C**** WILL THE USER WAIT FOR A TERMINAL?
       RN=DRAND(1)
       II=NNQ(ILOCAT+ITRMOS)+1
       IF (II .GT. MAXWAT) II=MAXWAT
       IF (RN .LE. TRMWAT(ILOCAT,II)) THEN
C          **** WILL WAIT ****
           WAITTR=1.0
           IF (LOGFLG) CALL LOG(15,-1)
```

```
      ELSE
C         **** WON'T WAIT -SCHEDULE LATER ARRIVAL ****
          WAITTR=0.0
          XTIME=RNORM(TRMLTM,TRMLTS,1)
          CALL SCHDL(LATEVT,XTIME,A)
          IF (LOGFLG) CALL LOG(16,-1)
      ENDIF
C
      RETURN
      END
C*************************************************************************
C                                                                        *
C  Name: XIOSEC                                                         *
C  Module Number: 1.F.9                                                 *
C  Function: Returns I/O seconds required as a function of class        *
C            and size.                                                   *
C  Attributes Referenced: none                                          *
C  Global (XX) Variables Referenced: none                               *
C  Common Blocks/Variables Used: FACTOR/DISTRB                          *
C  Common Blocks/Variables Changed: none                                *
C  Modules Called: none                                                 *
C  Calling Modules: CPUARV                                              *
C  Scheduled by: none                                                   *
C                                                                        *
C*************************************************************************
C
      FUNCTION XIOSEC(ICLASS,ISIZE)
      PARAMETER (MAXCLS=6,MAXSIZ=6)
C
C**** COMMON BLOCKS
      COMMON/FACTOR/ DISTRB(MAXCLS,MAXSIZ,6,2)
C
C**** WHAT IS THE I/O SECONDS REQUIRED
      XMEAN=DISTRB(ICLASS,ISIZE,6,1)
      STD=DISTRB(ICLASS,ISIZE,6,2)
      XIOSEC=RNORM(XMEAN,STD,1)
      IF (XIOSEC .LT. 10.) XIOSEC=10.
      RETURN
      END
```

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/OS/83D-1	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) See Box 19	
12. PERSONAL AUTHOR(S) McDermott, Steven Mark B.S., Capt, USAF	
13a. TYPE OF REPORT MS Thesis 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1983 December 15. PAGE COUNT 279
16. SUPPLEMENTARY NOTATION	Approved for public release: IAW AFR 190-17 ERIN E. WOLAVER Dean for Research and Professional Development Air Force Institute of Technology (ATC) Wright-Patterson AFB OH 45433

17. COSATI CODES		
FIELD	GROUP	SUB GR.
09	02	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Computer Models, Computer Network Models, Computer Workload Models, Simulation, SLAM, Workload Characterization

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

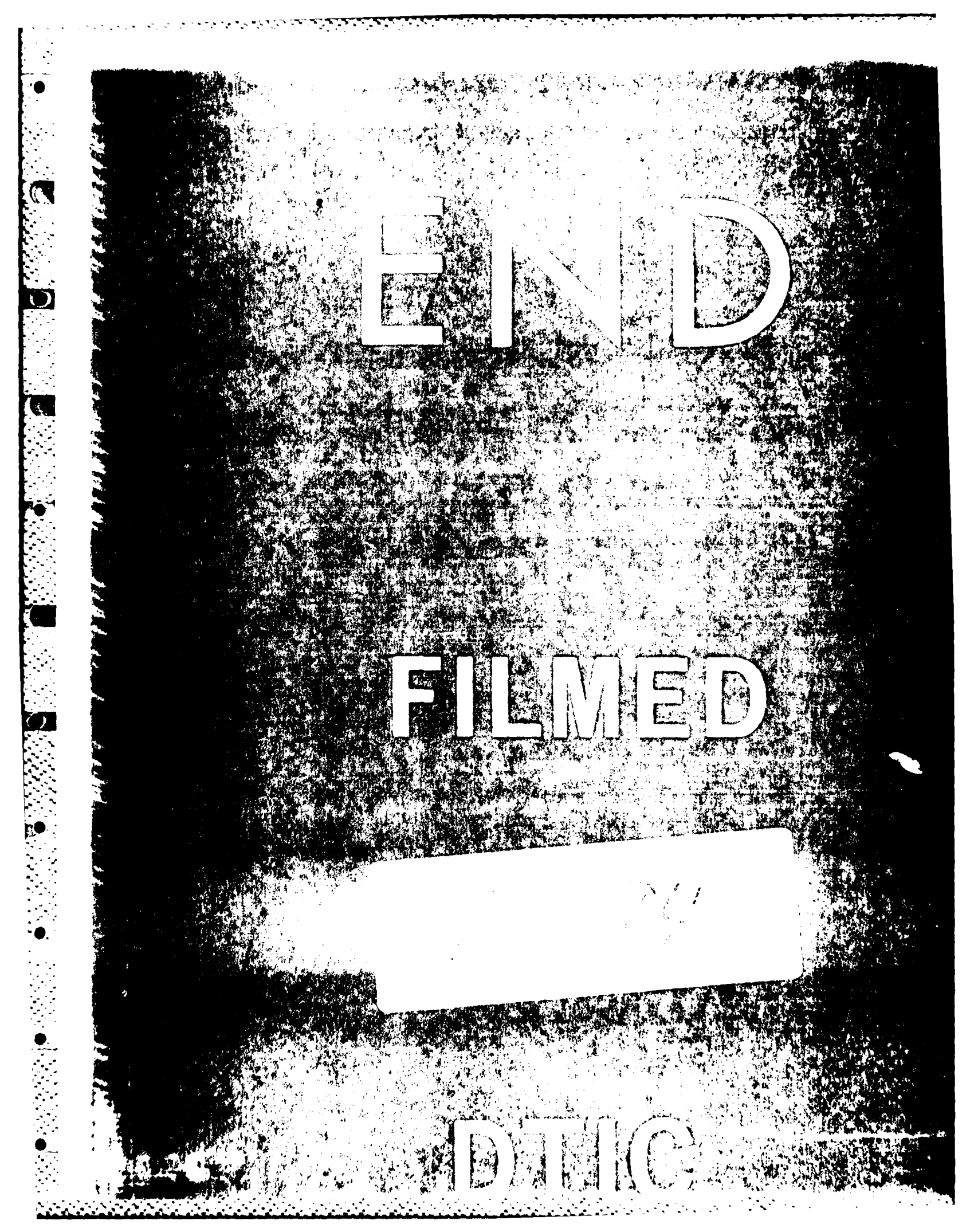
Title: DEVELOPMENT OF AN AFIT ADP SYSTEM NETWORK MODEL

Thesis Chairman: Thomas C. Clark, Lt Col, USAF

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Thomas C. Clark, Lt Col, USAF	22b. TELEPHONE NUMBER (Include Area Code) 513-255-3362 22c. OFFICE SYMBOL AFIT/ENS

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

A dynamic simulation model of the AFIT computer network system has been developed and tested. The model provides a broad-based structure of the AFIT network that can be used to evaluate different hardware configurations and models-- a workload and network model. The FORTRAN workload model transforms input requirement descriptions into a characterization of the workload capable of driving the network model. Using this characterization, the SLAM network model simulates the utilization of the AFIT network. The network model may be easily configured for any network. User guides, maintenance guides, and source code listings are provided.


END
FILMED
DTIC